ROHM GROUP

LAPIS SEMICONDUCTOR

FEUL-U16-100-INST-03

nX-U16/100 Core
Cẩm nang hướng dẫn

Vi điều khiển CMOS 16-bit

Ngày phát hành: Tháng 1. 2015

ROHM SEMICONDUCTOR

NOTES

Không được phép sao chép hoặc tái bản tài liệu này, một phần hoặc toàn bộ, nếu không có sự đồng ý của LAPIS Semiconductor Co., Ltd. Nội dung được nêu ở đây có thể thay đổi để cải thiện mà không cần thông báo..

Ví dụ về mạch ứng dụng, hằng số mạch và bất kỳ thông tin nào khác có trong đây minh họa cách sử dụng và vận hành tiêu chuẩn của Sản phẩm. Các điều kiện ngoại vi phải được tính đến khi thiết kế mạch để sản xuất hàng loạt.

Chúng tôi đã rất cẩn thận trong việc đảm bảo tính chính xác của thông tin được nêu trong tài liệu này. Tuy nhiên, nếu bạn gặp phải bất kỳ thiệt hại nào phát sinh do bất kỳ sự không chính xác hoặc in sai thông tin đó, LAPIS Semiconductor sẽ không chịu trách nhiệm về thiệt hại đó.

Thông tin kỹ thuật được nêu ở đây chỉ nhằm mục đích hiển thị các chức năng điển hình và ví dụ về mạch ứng dụng cho Sản phẩm. LAPIS Semiconductor không cấp cho bạn, rõ ràng hoặc ngầm định, bất kỳ giấy phép nào để sử dụng hoặc thực hiện quyền sở hữu trí tuệ hoặc các quyền khác do LAPIS Semiconductor và các bên khác nắm giữ. LAPIS Semiconductor sẽ không chịu bất kỳ trách nhiệm nào đối với bất kỳ tranh chấp nào phát sinh từ việc sử dụng thông tin kỹ thuật đó.

Các Sản phẩm được chỉ định trong tài liệu này được thiết kế để sử dụng với thiết bị hoặc thiết bị điện tử sử dụng chung (như thiết bị nghe nhìn, thiết bị tự động hóa văn phòng, thiết bị truyền thông, thiết bị điện tử và thiết bị giải trí).

Các Sản phẩm được chỉ định trong tài liệu này không được thiết kế để chịu được bức xạ.

Mặc dù LAPIS Semiconductor luôn nỗ lực nâng cao chất lượng và độ tin cậy của Sản phẩm, nhưng Sản phẩm có thể bị hỏng hoặc trục trặc vì nhiều lý do.

Xin hãy đảm bảo thực hiện trong thiết bị của bạn các biện pháp an toàn Sản phẩm để bảo vệ chống lại khả năng thương tích về thể chất, hỏa hoạn hoặc bất kỳ thiệt hại nào khác xảy ra trong trường hợp bất kỳ Sản phẩm nào bị hỏng, chẳng hạn như giảm công suất, dự phòng, kiểm soát hỏa hoạn và thiết kế an toàn. LAPIS Semiconductor sẽ không chịu bất kỳ trách nhiệm nào đối với việc bạn sử dụng bất kỳ Sản phẩm nào ngoài phạm vi quy định hoặc không theo hướng dẫn sử dụng.

Sản phẩm không được thiết kế hoặc sản xuất để sử dụng với bất kỳ thiết bị, dụng cụ hoặc hệ thống nào đòi hỏi mức độ tin cậy cực kỳ cao mà sự cố hoặc trục trặc của chúng có thể dẫn đến mối đe dọa trực tiếp đến tính mạng con người hoặc tạo ra nguy cơ gây thương tích cho con người (chẳng hạn như dụng cụ y tế, thiết bị vận tải, máy móc hàng không vũ trụ, bộ điều khiển lò phản ứng hạt nhân, bộ điều khiển nhiên liệu hoặc thiết bị an toàn khác). LAPIS Semiconductor sẽ không chịu bất kỳ trách nhiệm nào đối với việc sử dụng bất kỳ Sản phẩm nào cho các mục đích đặc biệt nêu trên. Nếu Sản phẩm được dự định sử dụng cho bất kỳ mục đích đặc biệt nào như vậy, vui lòng liên hệ với đại diện bán hàng của ROHM trước khi mua.

Nếu bạn có ý định xuất khẩu hoặc vận chuyển ra nước ngoài bất kỳ Sản phẩm hoặc công nghệ nào được nêu tại đây có thể được kiểm soát theo Luật Ngoại hối và Luật Thương mại nước ngoài, bạn sẽ phải xin giấy phép hoặc giấy phép theo Luật.

Nội dung

4. Phụ lục 4-1

1. Kiến trúc

1.1 Tổng quan

CPU nX-U16/100 có khả năng thực hiện các lệnh hiệu quả trên cơ sở một lệnh mỗi xung clock thông qua xử lý song song bằng kiến trúc dạng pipeline 3 giai đoạn.

nX-U16/100 được phân loại như sau dựa trên sự khác biệt về số chu kỳ thực hiện khi nhánh có điều kiện được kết thúc.

Loại lõi CPU	Chu kỳ thực hiện của lệnh rẽ nhánh có điều kiện.
A33 , A34	1/3 Điều kiện nhánh là (chưa đáp ứng/đáp ứng).
A35	1/2 Điều kiện nhánh là (chưa đáp ứng/đáp ứng).

Loại lõi CPU của mỗi sản phẩm là khác nhau. Để biết chi tiết, vui lòng tham khảo hướng dẫn sử dụng.

Khi loại CPU không được mô tả trong hướng dẫn sử dụng, lõi A34 sẽ được trang bị.

1.1.1 Tính năng

Kiến trúc U16 có các tính năng sau.

- **Bộ lệnh mạnh mẽ**
  Lệnh về truyền dữ liệu, số học, so sánh, phép toán logic, thao tác bit, phép toán logic bit, nhánh, nhánh có điều kiện, thao tác ngăn xếp gọi/trả về (call/return stack) và dịch chuyển số học
- **Nhiều chế độ định danh địa chỉ**
  Địa chỉ thanh ghi
  Địa chỉ thanh ghi gián tiếp
  Địa chỉ con trỏ ngăn xếp
  Địa chỉ thanh ghi điều
  khiển
  Địa chỉ thanh ghi gián tiếp EA
  Địa chỉ thanh ghi gián tiếp mục đích chung
  Địa chỉ trực tiếp
  Địa chỉ bit gián tiếp thanh ghi
  Địa chỉ bit trực tiếp
- **Không gian bộ nhớ**
  -Bộ nhớ chương trình/mã (ROM)
  Tối đa 16 phân đoạn (seg), mỗi phân đoạn 32 kiloword (0000H-FFFFH)
  - Bộ nhớ dữ liệu (RAM)
  Tối đa 256 phân đoạn, mỗi phân đoạn 64 kilobyte (0000H-FFFFH)
- **Ngắt (Interrupts)**
  Ngắt trình giả lập chuyên
  dụng
  Ngắt không thể vô hiệu hóa
  Ngắt có thể vô hiệu hóa
  Ngắt phần mềm

1.2 Tài nguyên CPU và Mô hình lập trình

Kiến trúc U16 có hai không gian địa chỉ: 1 megabyte cho mã và 16 megabyte cho dữ liệu. Cả hai không gian địa chỉ đều được chia thành các phân đoạn vật lý, mỗi phân đoạn 64 kilobyte. Tuy nhiên, cấu hình bộ nhớ cho phân đoạn vật lý #0 (0:0000H đến 0:FFFFH) khác với các phân đoạn khác.

Phân đoạn vật lý #0 cung cấp hai bộ địa chỉ và các thanh ghi riêng biệt để truy cập chúng: một phân đoạn bộ nhớ chương trình/mã 32 kiloword được truy cập bằng bộ đếm chương trình (PC) và một phân đoạn bộ nhớ dữ liệu 64 kilobyte được truy cập bằng thanh ghi địa chỉ (AR). Tuy nhiên, nếu địa chỉ trong AR nằm trong cửa sổ ROM, thanh ghi sẽ truy cập bộ nhớ chương trình/mã, chứ không phải bộ nhớ dữ liệu.

Các phân đoạn vật lý từ #1 trở lên, với các địa chỉ trên 64 kilobyte đầu tiên, tạo thành một không gian địa chỉ duy nhất kết hợp bộ nhớ chương trình/mã và dữ liệu. Việc truy cập một phân đoạn vật lý được chỉ định cho bộ nhớ chương trình/mã yêu cầu một địa chỉ 20 bit (CSR:PC) kết hợp bốn bit từ thanh ghi phân đoạn mã (CSR) và 16 bit từ bộ đếm chương trình (PC); một phân đoạn vật lý được chỉ định cho bộ nhớ dữ liệu, một địa chỉ 24 bit (DSR:AR) kết hợp tám bit từ thanh ghi phân đoạn dữ liệu (DSR) và 16 bit từ thanh ghi địa chỉ (AR).

Hình 1.1 tóm tắt cách bố trí các không gian bộ nhớ U16.

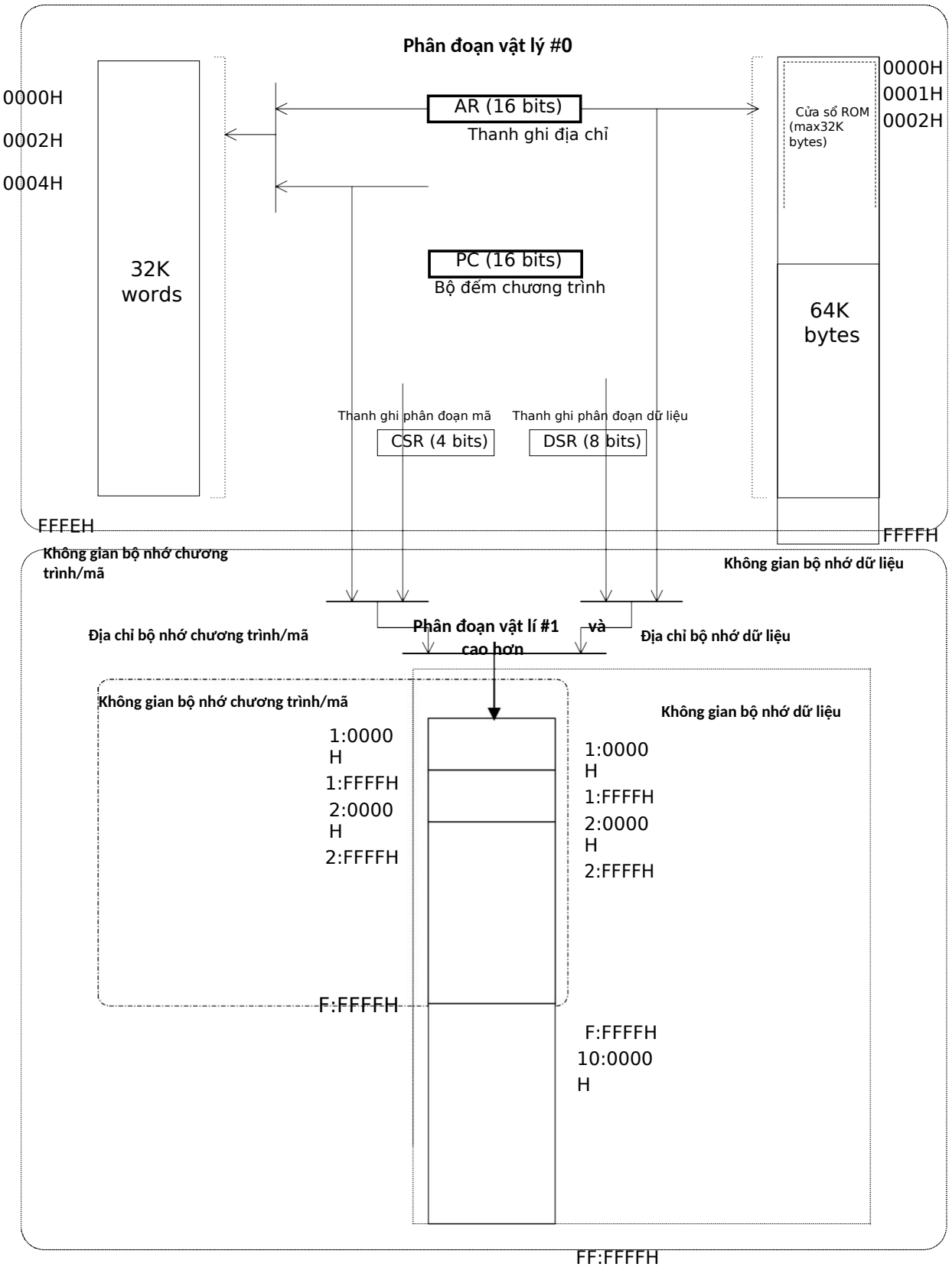


Figure 1.1. Không gian Bộ nhớ U16

1.2.1 Các thanh ghi

Các thanh ghi chung nằm ở trung tâm của hệ thống phần cứng U16. Cũng được hiển thị trong Hình 1.2 là các thanh ghi điều khiển.

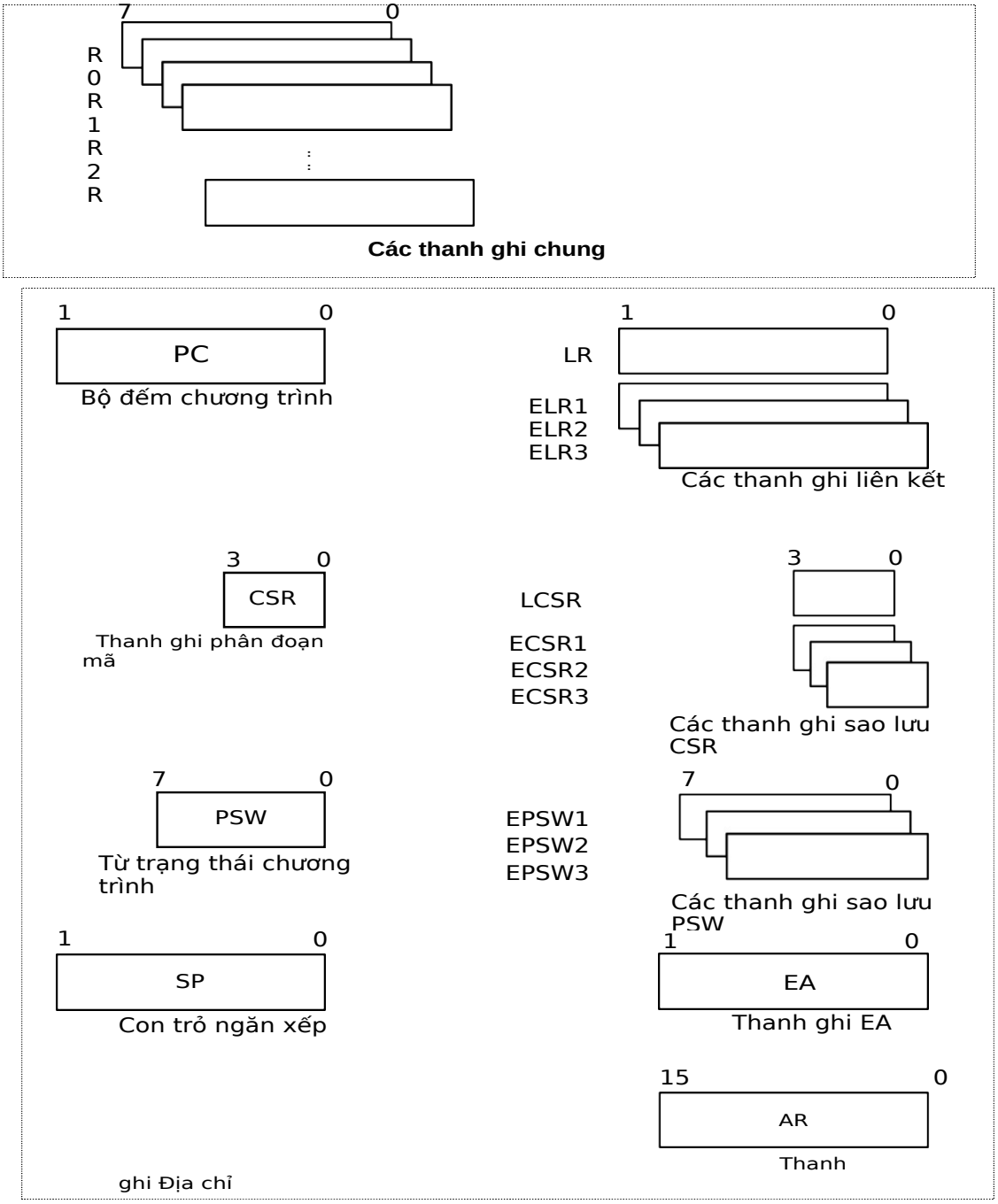


Figure 1.2. Register Set

1.2.1.1 Các thanh ghi chung

16 thanh ghi này ở trung tâm tính toán có chiều rộng một byte. Tuy nhiên, các chế độ địa chỉ đặc biệt cũng nhóm các thanh ghi liền kề lại với nhau để cho phép truy cập dưới dạng tám thanh ghi có kích thước từ (ERn), bốn thanh ghi có kích thước từ đôi (XRn) và hai thanh ghi có kích thước từ bốn (QRn).

Nếu trình xử lý ngắt sửa đổi nội dung của các thanh ghi này, nó phải lưu chúng một cách rõ ràng bằng lệnh PUSH tại điểm vào và khôi phục chúng bằng lệnh POP trước khi trả về..

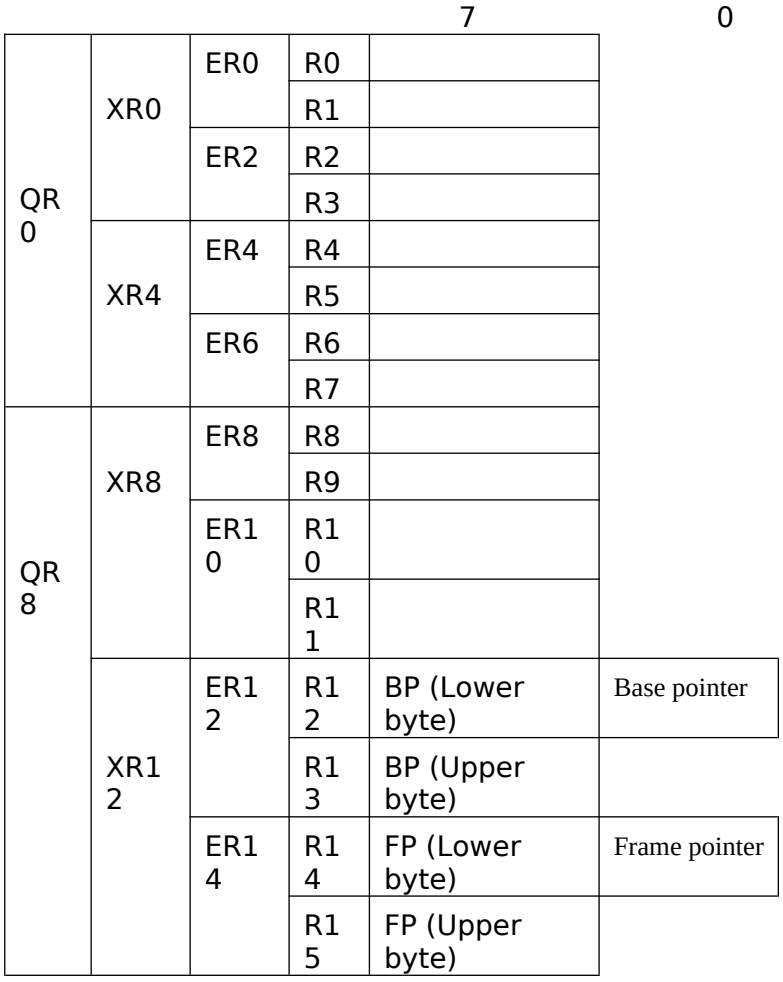


Figure 1.3. Các thanh ghi chung

Examples: Sử dụng thanh ghi chung

```
MOV   R0   ,   #7      ; thanh ghi có kích thước byte
L     ER0    , [EA+]   ; thanh ghi có kích thước word
L     XR0    , [EA]    ; thanh ghi có kích thước word gấp đôi
ST    QR0    , [EA]    ; thanh ghi có kích thước word gấp 4
```

```
SB      R3.2            ; bit riêng lẻ trong thanh
                        ghi
```

1.2.1.2 Con trỏ cơ sở và con trỏ khung (Base and Frame Pointers)

Trình biên dịch C sử dụng hai con trỏ toàn cục. Nó sử dụng ER12 làm con trỏ cơ sở (BP) và ER14 làm con trỏ khung (FP). Do đó, hai thanh ghi này cung cấp các chế độ địa chỉ đặc biệt ngoài vai trò là thanh ghi chung. Để biết thêm chi tiết, hãy xem Chương 2.

1.2.2 Thanh ghi điều khiển

Các thanh ghi này kiểm soát luồng chương trình và lưu giữ thông tin trạng thái hoạt động. Có 18 thanh ghi như vậy, mỗi thanh ghi có chức năng đặc biệt riêng. Nội dung của toàn bộ nhóm đôi khi được gọi là ngữ cảnh chương trình.

1.2.2.1 Thanh ghi trạng thái chương trình [Program Status Word (PSW)]

	7	6	5	4	3	2	1	0
PSW	C	Z	S	OV	MIE	HC	ELEVEL	

Thanh ghi 8 bit này chứa năm cờ (flag) theo dõi kết quả thực hiện lệnh, một bit điều khiển và một trường (field) dữ liệu.

Phần cứng tự động lưu những nội dung này vào thanh ghi trạng thái chương trình ngoại lệ (EPSW) khi nó chấp nhận yêu cầu ngắt. Lệnh RTI ở cuối trình xử lý ngắt sẽ khôi phục chúng.

Thanh ghi này chứa năm cờ (flag) theo dõi kết quả thực hiện lệnh số học, một bit điều khiển chấp nhận ngắt và một trường 2 bit chỉ ra mức ngoại lệ (ELEVEL). Chương trình có thể thay đổi các nội dung này bất kỳ lúc nào. Sau khi thiết lập lại, tất cả đều bằng không.

Các flag, bit và trường này có các chức năng sau.

- Bit 7: Cờ nhớ-Carry flag (C)
  Bit này sẽ chuyển thành “1” nếu lệnh số học, dịch chuyển hoặc so sánh tạo ra phép nhớ bit 7 hoặc bit 0 hoặc phép mượn vào bit 7. Nếu không, nó sẽ chuyển thành “0.”
  Nội dung cũng có thể được thiết lập, đặt lại và đảo ngược trực tiếp bằng các lệnh SC, RC hoặc CPLC và được kiểm tra bằng các lệnh nhánh có điều kiện.
- Bit 6: Cờ không-Zero flag (Z)
  Bit này sẽ chuyển thành “1” nếu lệnh số học hoặc truyền dữ liệu tạo ra kết quả bằng không. Nếu không, nó sẽ chuyển thành “0.”
  Nội dung có thể được kiểm tra bằng lệnh nhánh có điều kiện..
- Bit 5: Cờ dấu (Cờ âm)-Sign flag (S)
  Bit này theo dõi bit dấu trong kết quả từ lệnh số học, so sánh hoặc logic bit: “1” cho số âm, “0” cho số dương.
- Bit 4: Cờ tràn-Overflow flag (OV)
  Bit này sẽ chuyển thành “1” nếu một lệnh số học có dấu tạo ra một phép thực hiện hoặc một phép mượn vào bit 7—tức là, một kết quả không phù hợp với phạm vi bổ sung hai có sẵn. Nếu không, nó sẽ chuyển thành “0.”

- Bit 3: Bit cho phép ngắt chính - Master interrupt enable bit (MIE)
  Bit này là một mặt nạ kiểm soát việc chấp nhận các yêu cầu ngắt có thể vô hiệu hóa (MI). Đặt thành “1” cho phép các yêu cầu ngắt như vậy; “0” vô hiệu hóa chúng.

  Phần cứng tự động đặt bit này thành “0” khi chấp nhận một yêu cầu ngắt có thể vô hiệu hóa (MI). Nội dung cũng có thể được đặt hoặc đặt lại trực tiếp bằng lệnh EI và DI.
- Bit 2: Cờ phụ (Cờ mang nửa)-Half carry flag (HC)

  Bit này, được sử dụng trong phép tính BCD, sẽ chuyển thành “1” nếu lệnh tính toán hoặc so sánh tạo ra phép nhớ hoặc phép mượn vào bit 3 hoặc bit 11. Nếu không, nó sẽ chuyển thành “0.”
- Bits 1 and 0: Mức ngoại lệ-Exception level (ELEVEL)

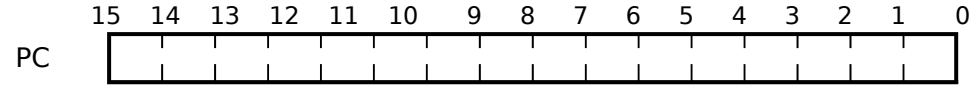
  Trường này cung cấp mức ngoại lệ hiện tại, một số nguyên từ 0 đến 3 biểu thị mức độ ưu tiên ngắt. Số này càng cao thì mức độ ưu tiên càng lớn. Để biết danh sách các ngắt và mức độ ngoại lệ/ưu tiên của chúng, hãy xem Mục 1.3.7 “Hoạt động ngắt”.

  Phần cứng U16 chỉ chấp nhận yêu cầu ngắt nếu mức ưu tiên ngắt của nó giống hoặc lớn hơn mức cài đặt mức ngoại lệ hiện tại (ELEVEL).

1.2.2.1.1 Lệnh sửa đổi cờ PSW

Để biết thêm chi tiết về Lệnh sửa đổi cờ PSW và bản chất chính xác của những sửa đổi đó, hãy xem Phần 3.2 “Lệnh theo Nhóm chức năng” và Chương 4 “Phụ lục.”

1.2.2.2 Bộ đếm chương trình [Program Counter (PC)]



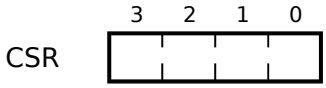
Thanh ghi 16 bit này giữ phần bù của địa chỉ của lệnh tiếp theo được thực thi. Phần cứng tự động tăng nó ngay sau khi lấy lệnh từ bộ nhớ chương trình/mã, tạo ra chu kỳ cần thiết để thực thi tuần tự. Tuy nhiên, lệnh nhánh và các lệnh khác phá vỡ chu kỳ này bằng cách ghi đè mặc định này bằng một địa chỉ khác.

Lệnh luôn bắt đầu ở ranh giới từ (word boundaries), do đó phần cứng tăng bộ đếm chương trình (PC) lên hai mỗi lần và buộc bit thấp nhất trong bất kỳ địa chỉ nào được tải thành "0" để thực thi sự căn chỉnh này.

Sau khi Reset, bộ đếm chương trình (PC) bắt đầu với nội dung của vectơ tương ứng với hệ số reset (reset factor).

Khi phần cứng chấp nhận yêu cầu ngắt, nó sẽ tự động lưu nội dung của thanh ghi này để sử dụng như một phần của địa chỉ trả về trong thanh ghi liên kết ngoại lệ (ELR1 đến ELR3) cho thiết lập mức ngoại lệ hiện tại (ELEVEL). Lệnh RTI ở cuối trình xử lý ngắt sẽ khôi phục chúng.

1.2.2.3 Thanh ghi phân đoạn mã [Code Segment Register (CSR)]



Thanh ghi 4 bit này giữ phần số phân đoạn vật lý (0 đến 15) của địa chỉ cho lệnh hiện tại. 16 bit còn lại (0 đến FFFFH), biểu diễn một offset trong phân đoạn vật lý đó, đến từ bộ đếm chương trình (PC). Cùng nhau, hai thanh ghi này chỉ định một địa chỉ 20 bit (CSR:PC) truy cập toàn bộ không gian bộ nhớ chương trình/mã.

Tính toán địa chỉ chỉ áp dụng cho offset 16 bit, bỏ qua bất kỳ tràn hoặc thiếu nào, do đó không bao giờ sửa đổi nội dung CSR. Điều tương tự cũng áp dụng cho tràn PC. Do đó, việc thực thi chương trình liên tục tuần hoàn qua các địa chỉ trong cùng một phân đoạn vật lý cho đến khi chương trình ghi đè rõ ràng nội dung CSR.

Các hành động sau đây sửa đổi nội dung CSR.

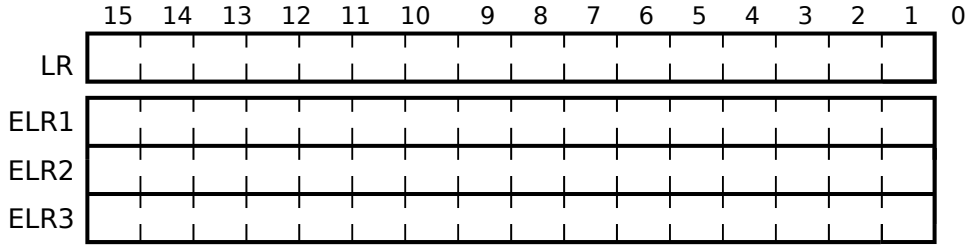
- chấp nhận ngắt: CSR về 0.
- reset : CSR về 0.
- Lệnh B *Cadr* : CSR đi đến giá trị được chỉ định trong lệnh.
- Lệnh BL *Cadr* : CSR đi đến giá trị được chỉ định trong lệnh.
- Lệnh RTI: CSR chuyển đến giá trị từ thanh ghi ECSR tương ứng với cài đặt mức ngoại lệ hiện tại (ELEVEL) từ thanh ghi trạng thái chương trình (PSW).

- Lệnh RT : CSR đi đến giá trị từ thanh ghi LCSR.
- Lệnh POP PC: CSR đi đến giá trị từ ngăn xếp.

Khi phần cứng chấp nhận yêu cầu ngắt, nó sẽ tự động lưu nội dung của thanh ghi này để sử dụng như một phần của địa chỉ trả về trong thanh ghi ECSR (ECSR1 đến ECSR3) cho thiết lập mức ngoại lệ hiện tại (ELEVEL). Lệnh RTI ở cuối trình xử lý ngắt sẽ khôi phục chúng.

Sau khi reset, thanh ghi này chứa không.

1.2.2.4 Thanh ghi liên kết (Link Registers) (LR, ELR1, ELR2, and ELR3)



Bốn thanh ghi 16 bit này dùng để lưu nội dung của bộ đếm chương trình (PC) trong các chương trình con (LR) và trình xử lý ngắt (ELR1 đến ELR3). Bit thấp nhất luôn là “0.”

Thanh ghi LR giữ phần bù của địa chỉ trả về cho một chương trình con được gọi bằng lệnh BL.

Lệnh RT ở cuối chương trình con tải nội dung LR trở lại bộ đếm chương trình (PC).

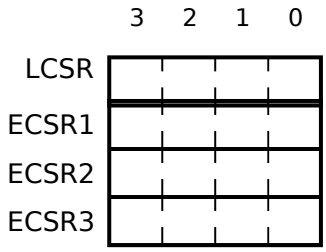
Lưu ý rằng chương trình có hai lựa chọn để trả về từ một chương trình con đến đến chương trình gọi nó: RT hoặc POP. Để biết thêm chi tiết, hãy xem Mục 1.4 “Mức độ ngoại lệ và Thanh ghi dự phòng.”

Các thanh ghi ELR1 đến ELR3 giữ các phần bù của địa chỉ trả về cho trình xử lý ngắt ở các mức ngoại lệ tương ứng. Phần cứng lưu địa chỉ trả về bằng cách sử dụng số chỉ mục được gán cho ngắt đang được chấp nhận. Để biết danh sách các ngắt và mức ngoại lệ/ưu tiên của chúng, hãy xem Mục 1.3.7 “Hoạt động ngắt.”

Lưu ý rằng việc sửa đổi phần ELEVEL của PSW trong phần mềm cần đặc biệt cẩn thận vì nó thay đổi chỉ mục trỏ đến cặp thanh ghi ELR-ECSR được sử dụng gần nhất.

Cũng lưu ý rằng cặp thanh ghi ELR3-ECSR3 chỉ có mặt vật lý trong các mô hình bao gồm trình gỡ lỗi trên chip. Truy cập các thanh ghi này trên các mô hình khác dẫn đến hoạt động không thể đoán trước. Phải luôn kiểm tra Hướng dẫn sử dụng cho thiết bị mục tiêu trước.

1.2.2.5 Thanh ghi sao lưu CSR (LCSR, ECSR1, ECSR2, and ECSR3)



Bốn thanh ghi 4 bit này dùng để lưu nội dung của thanh ghi phân đoạn mã (CSR) trong các chương trình con (LCSR) và trình xử lý ngắt (ECSR1 đến ECSR3).

Thanh ghi LCSR giữ phần phân đoạn vật lý của địa chỉ trả về cho một chương trình con được gọi bằng lệnh BL. Lệnh RT ở cuối chương trình con tải nội dung LCSR trở lại thanh ghi phân đoạn mã (CSR).

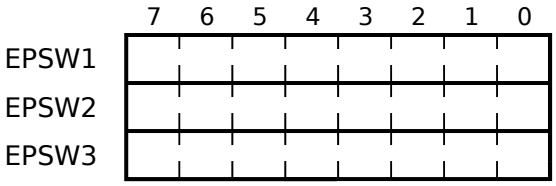
Lưu ý rằng chương trình có hai lựa chọn lệnh để trả về từ một chương trình con cho chương trình gọi nó: RT hoặc POP. Để biết thêm chi tiết, hãy xem Mục 1.4 “Mức độ ngoại lệ và Thanh ghi dự phòng.”

Các thanh ghi ECSR1 đến ECSR3 giữ các phần phân đoạn vật lý của địa chỉ trả về cho trình xử lý ngắt ở các mức ngoại lệ tương ứng. Phần cứng lưu địa chỉ trả về bằng cách sử dụng số chỉ mục được gán cho ngắt đang được chấp nhận. Để biết danh sách các ngắt và mức ngoại lệ/ưu tiên của chúng, hãy xem Phần 1.3.7 “Hoạt động ngắt”.

Lưu ý rằng việc sửa đổi phần ELEVEL của PSW trong phần mềm cần đặc biệt cẩn thận vì nó thay đổi chỉ mục trỏ đến cặp thanh ghi ELR-ECSR được sử dụng gần nhất.

Cũng lưu ý rằng cặp thanh ghi ELR3-ECSR3 chỉ có mặt vật lý trong các mô hình bao gồm trình gỡ lỗi trên chip. Truy cập các thanh ghi này trên các mô hình khác dẫn đến hoạt động không thể đoán trước. Luôn kiểm tra Hướng dẫn sử dụng cho thiết bị mục tiêu trước.

1.2.2.6 Thanh ghi sao lưu trạng thái chương trình là gì (EPSW1, EPSW2, and EPSW3)



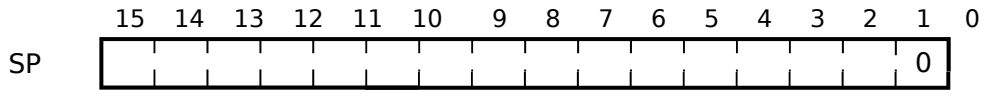
Ba thanh ghi 8 bit này có chức năng lưu nội dung của PSW trong quá trình xử lý ngắt.

Phần cứng lưu PSW bằng cách sử dụng số chỉ mục được gán cho ngắt đang được chấp nhận. Để biết danh sách các ngắt và mức độ ngoại lệ/ưu tiên của chúng, hãy xem Phần 1.3.7 “Hoạt động ngắt”.

Lưu ý rằng việc sửa đổi phần ELEVEL của PSW trong phần mềm cần đặc biệt cẩn thận vì nó thay đổi chỉ mục trỏ đến thanh ghi EPSW được sử dụng gần nhất.

Cũng lưu ý rằng thanh ghi EPSW3 chỉ có mặt vật lý trong các mô hình bao gồm trình gỡ lỗi trên chip. Truy cập thanh ghi này trên các mô hình khác dẫn đến hoạt động không thể đoán trước. Luôn kiểm tra Hướng dẫn sử dụng cho thiết bị mục tiêu trước.

1.2.2.7 Con Trỏ Ngăn Xếp (SP)



Thanh ghi 16 bit này chứa một con trỏ đến đầu ngăn xếp để lưu và khôi phục nội dung của các thanh ghi —ví dụ như với các lệnh PUSH và POP.

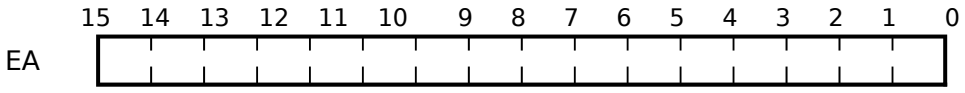
Các hoạt động ngăn xếp luôn có kích thước word. Một lần lưu dữ liệu có kích thước word vào ngăn xếp sẽ trừ 2 khỏi thanh ghi này rồi sao chép dữ liệu vào địa chỉ mới đó. Việc khôi phục dữ liệu sẽ sao chép một word từ ngăn xếp vào đích đã chỉ định rồi thêm 2 vào thanh ghi này.

Bit 0 của thanh ghi này được nối cứng với bit 0.

Thanh ghi này là thanh ghi độc lập, có thể truy cập hoàn toàn từ các chương trình có lệnh thích hợp—ví dụ như PUSH và POP.

Sau khi reset, thanh ghi này chứa nội dung của các địa chỉ 0000H và 0001H trong bộ nhớ chương trình/mã theo byte thấp và byte cao tương ứng.

1.2.2.8 Thanh ghi EA (EA)

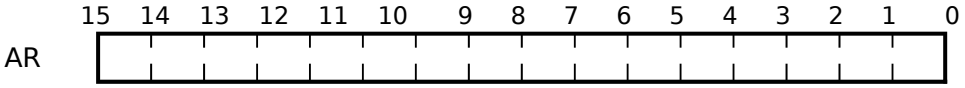


Thanh ghi 16 bit này lưu trữ một địa chỉ để các lệnh truy cập bộ nhớ dữ liệu gián tiếp thông qua thanh ghi này sử dụng.

16 bit này đủ để truy cập các địa chỉ bộ nhớ dữ liệu trong phân đoạn vật lý #0. Tuy nhiên, để truy cập các phân đoạn vật lý #1 trở lên, cần phải thêm tiền tố vào phần bù này bằng nội dung của thanh ghi phân đoạn dữ liệu (DSR), được mô tả bên dưới, để tạo thành địa chỉ 24 bit (DSR:EA).

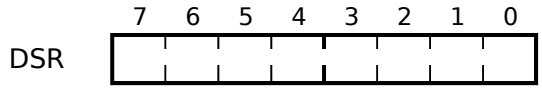
Có thể truy cập thanh ghi này từ các chương trình bằng lệnh LEA để tải thanh ghi này và bằng lệnh thao tác ngăn xếp PUSH và POP.

1.2.2.9 Thanh ghi địa chỉ (AR)



Thanh ghi 16 bit này tạm thời giữ một địa chỉ để sử dụng bởi các lệnh truy cập bộ nhớ dữ liệu. Nó dành riêng cho lõi U16 sử dụng, do đó không thể truy cập từ các chương trình.

1.2.2.10 Thanh ghi phân đoạn dữ liệu (DSR)



Thanh ghi 8 bit này giữ một số phân đoạn vật lý để truy cập bộ nhớ dữ liệu trong các phân đoạn vật lý #1 trở lên. Số này có thể nằm trong khoảng từ 0 đến 255.

Truy cập các địa chỉ trong phân đoạn vật lý được chỉ định sử dụng offset 16 bit (0 đến FFFFH) trong địa chỉ EA—tức là địa chỉ 24 bit với nội dung của thanh ghi này ở tám bit trên cùng và nội dung của thanh ghi EA ở 16 bit dưới cùng.

Các lệnh truy cập bộ nhớ chỉ định giá trị số cho phân đoạn vật lý trước tiên cập nhật DSR thành giá trị mới này. Các lệnh có ký hiệu DSR ở vị trí đó sử dụng nội dung hiện tại của thanh ghi này. Nếu không có ký hiệu nào, lệnh luôn bỏ qua nội dung DSR và sử dụng phân đoạn vật lý #0.

Đoạn mã sau đây đưa ra một số ví dụ về truy cập bộ nhớ dữ liệu.

```
   L       R0      ,5:1234H  ;      Đặt DSR thành 5 và tải R0 từ 5:1234H, một địa chỉ
   trong
                                    phân đoạn vật lý #1 trở lên.
;
   LEA             55AAH
   ST      R0      ,3:[EA+]  ;      Đặt DSR thành 3 và lưu trữ nội dung của R0 trong
   3:55AAH,
                                    một địa chỉ trong các phân đoạn vật lý #1 trở lên.
                             ;      Tăng EA.
   ST      R1      ,3:[EA+]  ;      Đặt DSR thành 3 và lưu trữ nội dung của R1 trong
   3:55ABH,
                                    một địa chỉ trong các phân đoạn vật lý #1 trở lên.
                             ;      Tăng EA.
   ST      R2      ,3:[EA+]  ;      Đặt DSR thành 3 và lưu trữ nội dung của R2 trong
   3:55ACH,
                                    một địa chỉ trong các phân đoạn vật lý #1 trở lên.
                             ;      Tăng EA.
;
   L       R0      ,5:1234H  ;      Đặt DSR thành 5 và tải R0 từ 5:1234H,một địa chỉ trong
   các
                                    phân đoạn vật lý #1 trở lên.
   L       R1      ,1234H    ;      Tải R1 từ offset 1234H trong phân đoạn vật lý bộ nhớ
   dữ liệu
                                    #0.
   L       R2      ,01235H   ;      Đặt DSR thành 0 và tải R2 từ offset 1235H trong phân
   đoạn
                                    vật lý bộ nhớ dữ liệu #0.
;
   LEA     AA55H             ;
   L       R5,DSR: [EA+]     ;      Tải R5 từ phân đoạn vật lý hiện có trong DSR bằng
                                    cách sử dụng offset trong EA (AA55H) .
                             ;      Tăng EA.
   L       R6,DSR:[EA+]      ;      Tải R6 từ phân đoạn vật lý hiện có trong DSR bằng
                                    cách sử dụng offset trong EA (AA56H).
                             ;      Tăng EA.
```

Sau khi reset, thanh ghi này chứa không.

1.3 Không gian bộ nhớ

Không gian bộ nhớ U16 bao gồm 256 phân đoạn vật lý, mỗi phân đoạn 64K byte. Nó được chia sẻ bởi không gian bộ nhớ chương trình/mã 1 megabyte (0:0000H đến F:FFFFH) và không gian bộ nhớ dữ liệu 16 megabyte (0:0000H đến FF:FFFFH). Tuy nhiên, phân đoạn vật lý #0 có cấu trúc khác với các phân đoạn khác, #1 trở lên.

1.3.1 Không gian bộ nhớ chương trình/mã

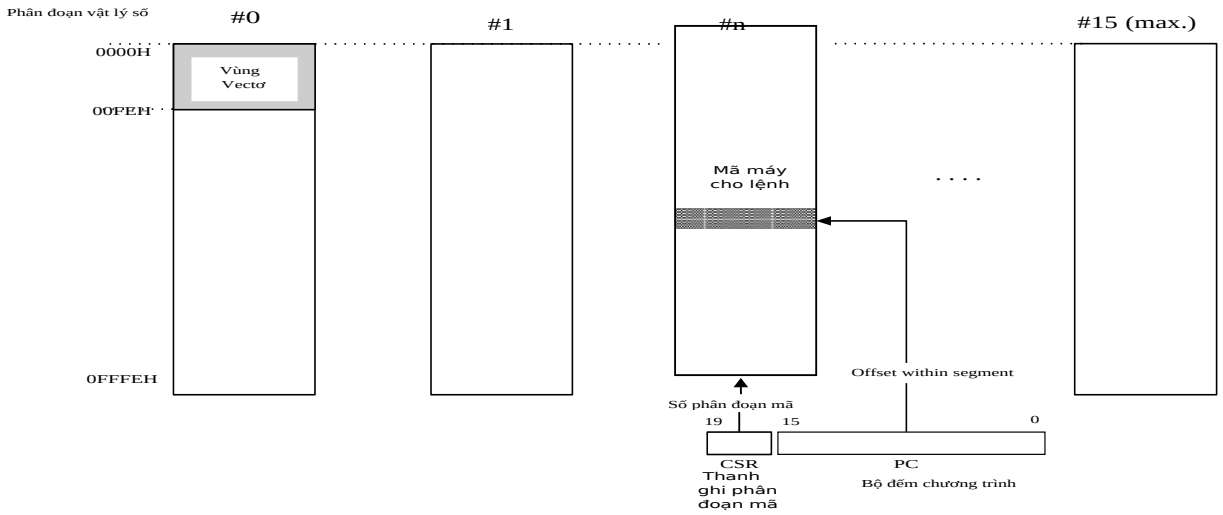
Không gian bộ nhớ chương trình/mã 1 megabyte của U16 có 16 phân đoạn vật lý với 32 kiloword mỗi phân đoạn. Công dụng chính của nó là lưu trữ mã máy cần thiết để thực thi chương trình và các bảng dữ liệu chỉ đọc. Các chương trình truy cập không gian này bằng địa chỉ 20 bit (CSR:PC) kết hợp nội dung của thanh ghi phân đoạn mã (CSR) ở bốn bit trên cùng và nội dung của bộ đếm chương trình (PC) ở 16 bit còn lại. Nội dung của thanh ghi phân đoạn mã (CSR) được gọi là phân đoạn mã.

Các tính toán địa chỉ như tăng bộ đếm chương trình (PC) và thêm hoặc trừ độ dịch chuyển để tính toán mục tiêu nhánh tương đối sẽ bỏ qua bất kỳ sự tràn hoặc thiếu nào, do đó không bao giờ sửa đổi nội dung CSR.

Cửa sổ ROM, một vùng đặc biệt trong phân đoạn vật lý #0, có thể truy cập bằng cách sử dụng địa chỉ RAM.

Truy cập bằng phân đoạn vật lý sử dụng offset 16 bit giữa 0 và 0FFFEH. Tính toán địa chỉ chỉ ảnh hưởng đến offset 16 bit này và bỏ qua bất kỳ tràn hoặc thiếu nào.

Sau đây minh họa cách bố trí không gian bộ nhớ này.



1.3.2 Bảng Vectơ

Địa chỉ 0:0H đến 0:0FEH trong không gian bộ nhớ chương trình/mã được dành riêng cho một bảng vectơ chứa các offset 16 bit cho các chương trình xử lý reset và ngắt. Mỗi vectơ trong bảng bắt đầu ở một địa chỉ chẵn. Phần cứng tự động đặt lại thanh ghi đoạn mã (CSR) về số không, vì vậy các chương trình này phải luôn bắt đầu ở phân đoạn vật lý #0.

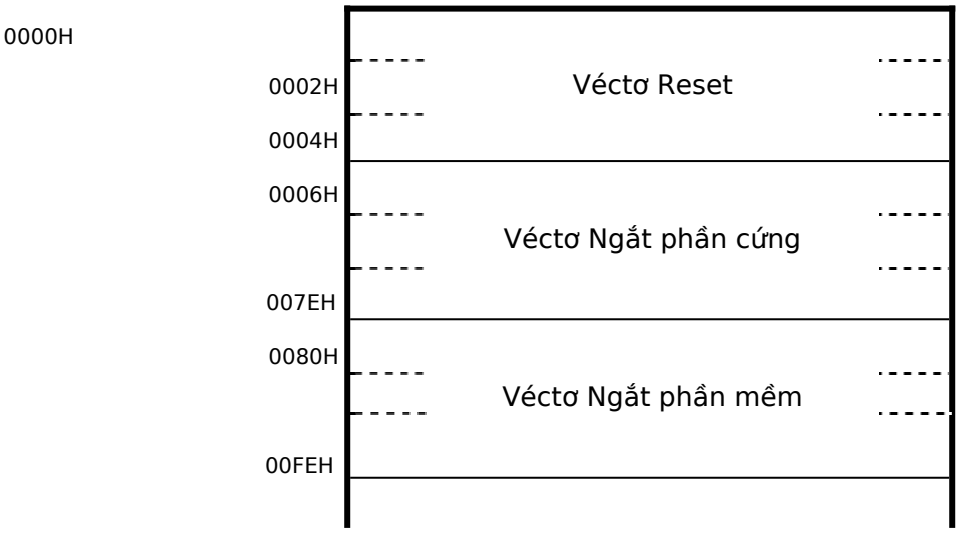
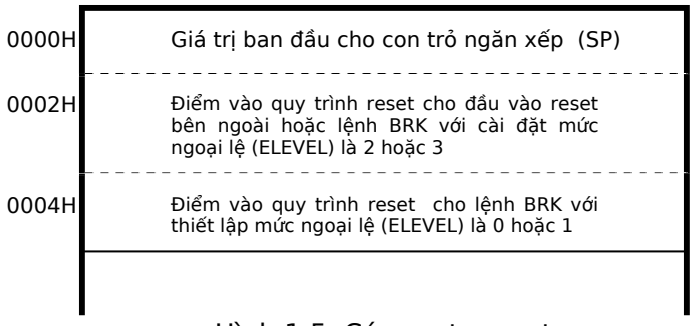


Figure 1.4. Bảng Vector

1.3.2.1 Vectơ Reset

Phần này của bảng vectơ chứa các điểm vào để xử lý reset—tức là giá trị ban đầu cho con trỏ ngăn xếp tại địa chỉ 0 và các điểm vào của trình tự reset tại địa chỉ 2 và 4.

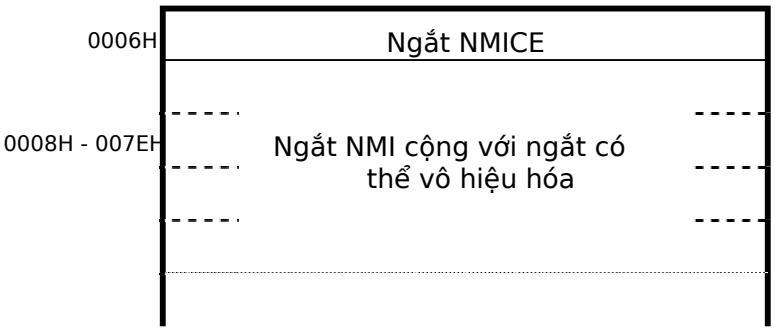


Hình 1.5. Các vectơ reset

1.3.2.2 Vectơ ngắt

1.3.2.2.1 Vectơ ngắt phần cứng

Phần này của bảng vector chứa các điểm vào để xử lý các ngắt phần cứng. Có hai yêu cầu ngắt không thể vô hiệu hóa, NMICE và NMI, cộng thêm chỗ cho tối đa 59 yêu cầu ngắt có thể vô hiệu hóa.



Hình 1.6. Các Vectơ Ngắt phần cứng

1.3.2.2.2 Vectơ ngắt phần mềm

Phần này của bảng vectơ giữ các điểm vào cho các yêu cầu ngắt từ các lệnh SWI trong chương trình.

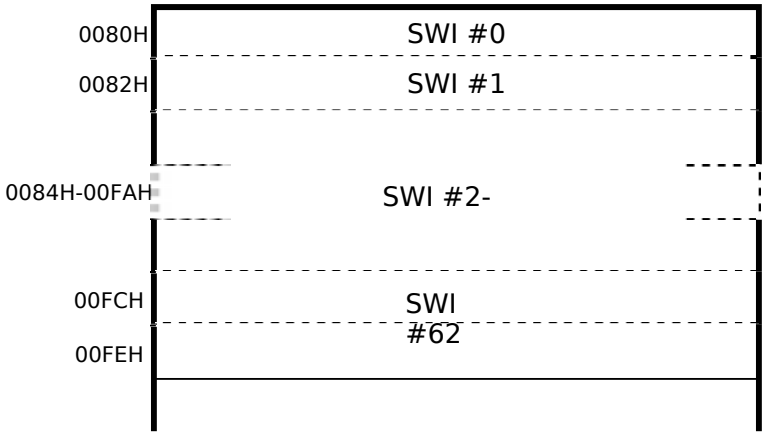


Figure 1.7. Vectơ ngắt phần mềm

1.3.2.3 Bảng Vectơ ghi

Trong ngôn ngữ lắp ráp, sử dụng các lệnh DW có nhãn biểu diễn các điểm vào làm toán hạng của chúng như được hiển thị trong đoạn mã sau.

Lưu ý rằng chỉ có các vectơ reset phải luôn có mặt. Nếu chương trình không sử dụng các ngắt này, vùng này có sẵn cho mã chương trình bình thường.

```
;
;reset vector
table
        cseg    at      0000h

        d       spini           ; Giá trị ban đầu cho con trỏ ngăn xếp
        w       t               ; Giá trị ban đầu cho bộ đếm chương
        d       start           trình
        w       brk             ; Điểm vào quá trình reset cho lệnh BRK
        d
        w

        or      0008h
        g       nmi_entry       ; Non-maskable interrupt
        dw
        d       Int1_entr       ; Maskable interrupt #0
        w       y               ; Maskable interrupt #1
;       d       Int2_entr
;software interrupts
        w       y
;               :
                :
                :


;
        cseg    at      0080h
;start of main procedure
swi_0:  dw      sw0_entry       ; Software interrupt
;                               #0
start:  dw      sw1_entry       ; Software interrupt
swi_1:                          #1
                :
                :
                :
```

1.3.3 Không gian bộ nhớ chương trình/mã

Theo quan điểm lập trình, không có sự khác biệt về mặt logic giữa phân đoạn vật lý #0 và các phân đoạn khác. Bộ liên kết và các công cụ khác tự động gán mã chương trình vào bộ nhớ tích hợp có sẵn trên chip và sau đó vào bộ nhớ ngoài.

1.3.4 Lệnh tiền tố DSR

Kiến trúc U16 chia không gian bộ nhớ thành các phân đoạn vật lý, mỗi phân đoạn có 64K byte, do đó việc truy cập dữ liệu trong một phân đoạn vật lý khác với phân đoạn vật lý #0 đòi hỏi phải thao tác thanh ghi phân đoạn dữ liệu (DSR) bằng một trong ba lệnh tiền tố DSR sau.

Lệnh tiền tố DSR	Chức năng
`1110_0011_iiii_iiii`	Tải DSR với giá trị tức thời 8 bit iiii_iiii.
`1001_0000_dddd_1111`	Tải DSR với nội dung của thanh ghi chung Rd.
`1111_1110_1001_1111`	Sử dụng giá trị DSR hiện tại.

Các lệnh tiền tố DSR chỉ có ảnh hưởng tiền tố này khi chúng đứng ngay trước một lệnh truy cập bộ nhớ. Các lệnh truy cập bộ nhớ không có lệnh tiền tố DSR đứng ngay trước đoạn vật lý truy cập #0.

Phần cứng tự động vô hiệu hóa tất cả các ngắt giữa lệnh tiền tố DSR và lệnh ngay sau đó. Để biết thêm chi tiết, hãy xem Phần 1.5 “Chặn ngắt” bên dưới.

Note: Để ngăn chặn hoạt động không mong muốn và cung cấp khả năng kiểm tra mạnh nhất có thể đối với quyền truy cập bộ nhớ, các thông số kỹ thuật ngôn ngữ lắp ráp U16 cố tình cấm sử dụng các lệnh tiền tố DSR trong mã nguồn chương trình. Thay vào đó, hãy sử dụng tiền tố DSR tương ứng bên trong chính lệnh truy cập bộ nhớ. Để biết thêm chi tiết, hãy xem Phần 2.3 “Địa chỉ bộ nhớ” bên dưới.

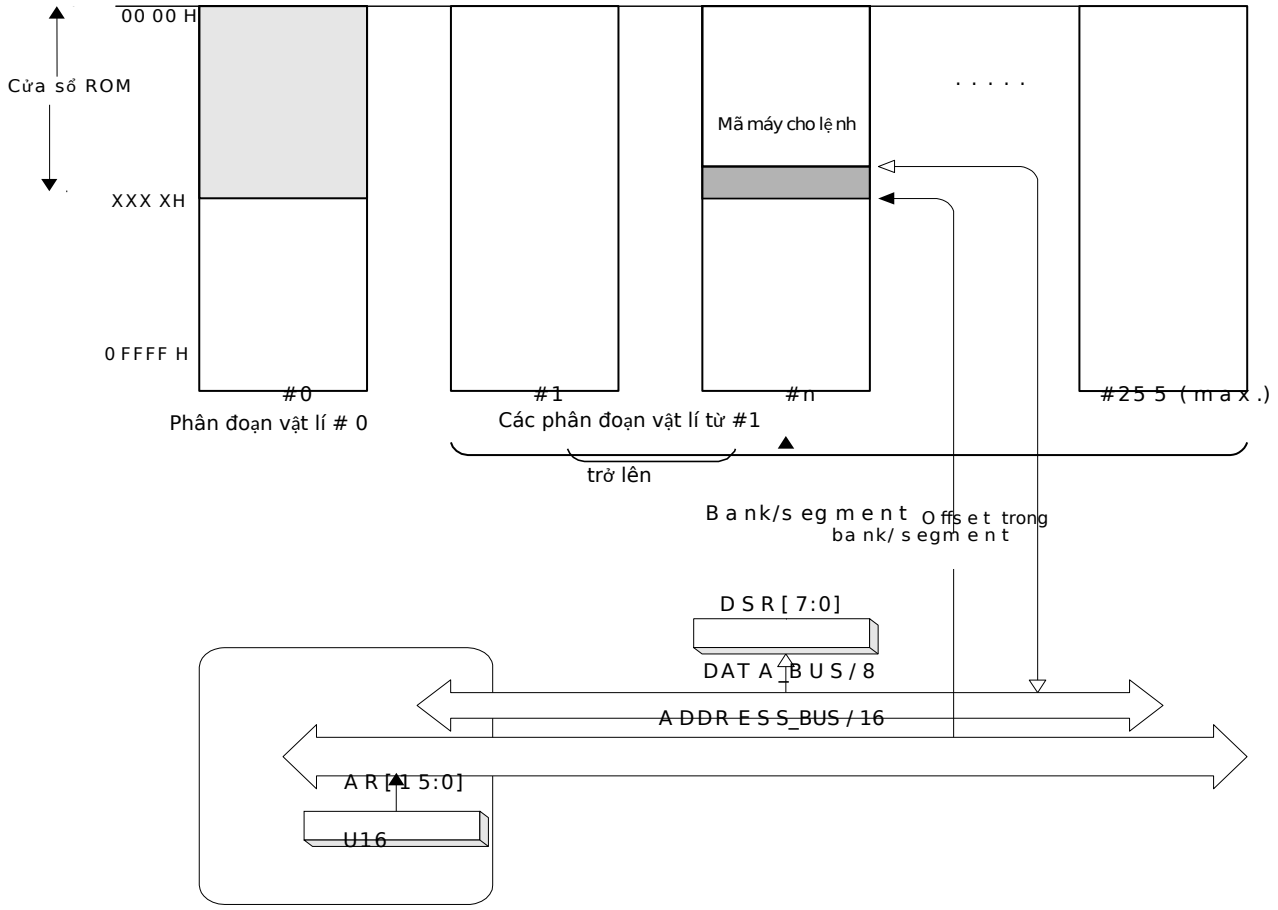
1.3.5 Không gian bộ nhớ dữ liệu

Không gian bộ nhớ dữ liệu 16 megabyte của U16 có 256 phân đoạn vật lý với 64 kilobyte mỗi phân đoạn. Công dụng chính của nó là lưu trữ dữ liệu được ghi cũng như đọc.

Các chương trình truy cập không gian này bằng địa chỉ 24 bit (DSR:AR) kết hợp nội dung của thanh ghi phân đoạn dữ liệu (DSR) trong tám bit trên cùng và nội dung của thanh ghi địa chỉ (AR) trong 16 bit còn lại. Nội dung của thanh ghi phân đoạn dữ liệu (DSR) được gọi là phân đoạn dữ liệu.

Phân đoạn vật lý #0 bao gồm cửa sổ ROM cộng với một hoặc hai vùng dữ liệu. Cửa sổ ROM là một vùng đặc biệt truy cập các địa chỉ bộ nhớ chương trình/mã bằng cách sử dụng địa chỉ RAM. Các địa chỉ bộ nhớ dữ liệu tương ứng không có mặt về mặt vật lý. Công dụng chính của cửa sổ này là truy cập dữ liệu bảng trong ROM.

Sau đây minh họa cách bố trí không gian bộ nhớ này.



1.3.5.1 Kiểu dữ liệu

Phần này mô tả các kiểu dữ liệu được hỗ trợ bởi các lệnh của U16.

Byte không dấu

Kiểu dữ liệu này được sử dụng bởi các lệnh hoạt động trên byte. Giá trị nằm trong khoảng từ 0 đến 255. Các phép toán số học tràn hoặc dưới phạm vi này sẽ đặt cờ nhớ (C) thành “1” và loại bỏ tất cả ngoại trừ tám bit thấp nhất để tạo ra kết quả modulo 256.

Các hoạt động bit thao tác các bit riêng lẻ. Các bit được đánh số từ 0 đến 7 từ bit ít quan trọng nhất (LSB) đến bit quan trọng nhất (MSB).

Byte có dấu

Kiểu dữ liệu này được sử dụng bởi các lệnh hoạt động trên byte. Bit trên cùng được coi là bit dấu, tạo ra các giá trị bù hai trong phạm vi từ -128 đến +127. Các phép toán số học tràn dưới hoặc tràn trên phạm vi này sẽ đặt cờ tràn (OV) thành “1.”

Word không có dấu

Kiểu dữ liệu này được sử dụng bởi các lệnh thao tác trên các word. Giá trị nằm trong khoảng từ 0 đến 65.535. Các phép toán số học tràn hoặc dưới phạm vi này sẽ đặt cờ nhớ (C) thành “1” và loại bỏ tất cả trừ 16 bit thấp nhất để tạo ra kết quả modulo 65.536.

Bộ nhớ lưu trữ là little endian, với byte thấp hơn (bit 7 đến 0) trước byte cao hơn (bit 15 đến 8). Bộ nhớ dữ liệu yêu cầu căn chỉnh ranh giới word với byte thấp hơn ở địa chỉ chẵn và byte cao hơn ở địa chỉ lẻ tiếp theo. Bộ nhớ chương trình/mã không áp đặt hạn chế này. Địa chỉ của dữ liệu word luôn là địa chỉ của byte thấp hơn của nó.

Các hoạt động bitwise thao tác các bit riêng lẻ. Các bit được đánh số từ 0 đến 15 từ bit ít quan trọng nhất (LSB) đến bit quan trọng nhất (MSB).

Word có dấu

Kiểu dữ liệu này được sử dụng bởi các lệnh hoạt động trên các word. Bit trên cùng được coi là bit có dấu, tạo ra các giá trị bù hai trong phạm vi từ -32768 đến + 32767. Các phép toán số học tràn dưới hoặc tràn trên phạm vi này sẽ đặt cờ tràn (OV) thành “1.”

Bộ nhớ lưu trữ là little endian, với byte thấp hơn (bit 7 đến 0) trước byte cao hơn (bit 15 đến 8). Bộ nhớ dữ liệu yêu cầu căn chỉnh ranh giới word với byte thấp hơn ở địa chỉ chẵn và byte cao hơn ở địa chỉ lẻ tiếp theo. Bộ nhớ chương trình/mã không áp đặt hạn chế này. Địa chỉ của dữ liệu word luôn là địa chỉ của byte thấp hơn của nó.

Bit

Kiểu dữ liệu này được sử dụng bởi các lệnh hoạt động trên bit. Các giá trị duy nhất là “0” và “1”. Kiểu này áp dụng cho từng bit trong hầu hết các thanh ghi và tất cả các bit trong bộ nhớ. Địa chỉ bit sử dụng tên của một thanh ghi có kích thước byte hoặc một địa chỉ bộ nhớ cộng với toán tử dấu chấm và số (0 đến 7). Các hoạt động có sẵn cho các bit này bao gồm chuyển, hoạt động logic , kiểm tra bit và nhảy.

1.3.5.2 Phân cấp địa chỉ

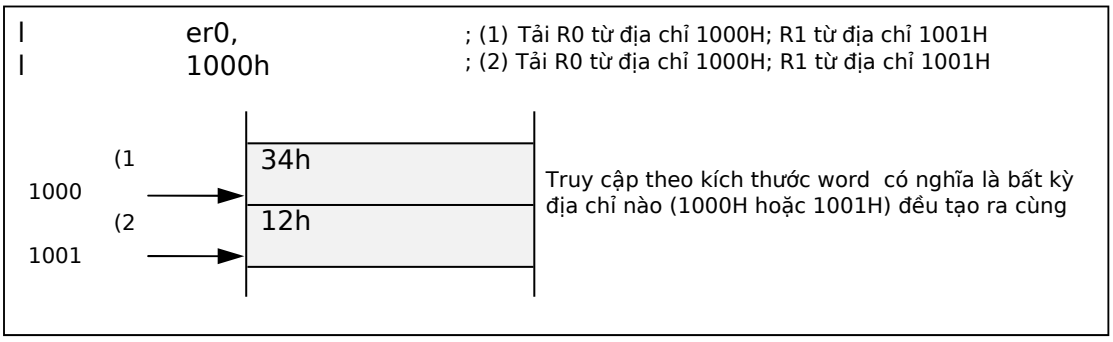
Địa chỉ bộ nhớ được tính bằng byte. Địa chỉ byte gán một địa chỉ duy nhất cho mỗi byte trong bộ nhớ. Các địa chỉ này chạy từ 0 đến FFFFH (65,535) trong mỗi phân đoạn vật lý 64 kilobyte.

Kiến trúc U16 phân tách bộ nhớ thành bộ nhớ chương trình/mã và bộ nhớ dữ liệu, mỗi bộ nhớ có một tập hợp địa chỉ byte riêng.

1.3.5.3 Ranh giới Word

Bộ nhớ dữ liệu U16 có ranh giới word. Phần cứng tự động thực thi căn chỉnh word bằng cách bỏ qua bit thấp nhất trong địa chỉ, buộc nó thành “0”. Các lệnh truy cập dữ liệu có kích thước word, double word hoặc quad word bằng cách sử dụng địa chỉ số lẻ do đó truy cập vào địa chỉ số chẵn trước đó mà không kích hoạt lỗi định danh địa chỉ. Do đó, lập trình viên phải gán các kiểu dữ liệu lớn hơn này cho ranh giới word . Lưu ý rằng không có ranh giới bổ sung nào cho các kiểu multiword . Yêu cầu duy nhất là chúng phải nằm trên ranh giới word.

Bộ nhớ chương trình/mã cũng có ranh giới word . Bộ nhớ chương trình/mã được truy cập bởi cửa sổ ROM cũng vậy.



Hình 1.8. Ranh giới word trong bộ nhớ

1.3.5.4 Cửa sổ ROM

Cửa sổ này, được gán cho một phần chưa sử dụng của phân đoạn vật lý #0 trong bộ nhớ dữ liệu, dùng để truy cập các địa chỉ bộ nhớ chương trình/mã tương ứng với địa chỉ RAM. Do đó, kiến trúc U16 không cần các lệnh đặc biệt để truy cập dữ liệu trong bộ nhớ chương trình/mã (ROM). Sử dụng lệnh truy cập RAM với địa chỉ trong cửa sổ ROM cũng tạo ra kết quả tương tự.

Tuy nhiên, việc truy cập một địa chỉ trong cửa sổ ROM mất nhiều chu kỳ lệnh hơn so với cùng một lệnh truy cập bộ nhớ dữ liệu. Để biết thêm chi tiết, hãy xem Phần 3.3 “Thời gian thực thi lệnh”.

Cửa sổ ROM chỉ hỗ trợ truy cập đọc. Ghi vào địa chỉ cửa sổ ROM không tạo ra kết quả có ý nghĩa.

1.3.6 Mô hình bộ nhớ phần cứng

Kiến trúc U16 cung cấp khả năng kiểm soát phần cứng đối với số lượng phân đoạn vật lý có thể truy cập được dưới dạng bộ nhớ chương trình/mã: 64 kilobyte (32 kiloword) hoặc 1 megabyte (512 kiloword). Quy trình chỉ định mô hình bộ nhớ phần cứng này có trong Sổ tay hướng dẫn sử dụng cho thiết bị mục tiêu.

Bảng sau đây tóm tắt các mô hình có sẵn.

Tên mô hình	Địa chỉ chương trình/mã	CSR Thanh ghi sao lưu CSR
SMALL	Bộ nhớ chương trình:0H–FFFFH Bộ nhớ dữ liệu:0H - FF:FFFFH	Không sử dụng
LARGE	Bộ nhớ chương trình: 0H - F:FFFFH Bộ nhớ dữ liệu: 0H - FF:FFFFH	Sử dụng

Việc lựa chọn mô hình bộ nhớ phần cứng ảnh hưởng đến các khía cạnh hoạt động sau:.

- Lượng bộ nhớ chương trình/mã khả dụng
- Hoạt động của các lệnh gọi chương trình con và các lệnh RT tương ứng
- Hoạt động của các lệnh ngắt và lệnh RTI tương ứng
- Hoạt động của lệnh PUSH và POP

Bảng sau đây phác thảo những khác biệt này chi tiết hơn.

	SMALL	LARGE
Lượng bộ nhớ chương trình/mã khả dụng	64K bytes (0H - 0FFFFH)	1M bytes (0H - F:FFFFH)
Các thanh ghi được lưu trong khi gọi chương trình con	PC	PC CSR
Các thanh ghi được khôi phục bằng lệnh RT	PC	PC CSR
Các thanh ghi được lưu trong quá trình chấp nhận ngắt	PC PSW	PC PSW CSR
Các thanh ghi được khôi phục bằng lệnh RTI	PC PSW	PC PSW CSR
Các thanh ghi được lưu bởi lệnh PUSH LR/ELR	LR	LR LCSR
	ELR	ELR ECSR
Các thanh ghi được khôi phục bằng lệnh POP LR/PC	LR	LR LCSR
	PC	PC CSR

1.3.7 Hoạt động ngắt

1.3.7.1 Chấp nhận ngắt

Phần cứng U16 chỉ chấp nhận yêu cầu ngắt (NMICE, NMI hoặc MI) nếu mức ngắt của nó giống hoặc lớn hơn mức ngoại lệ hiện tại (ELEVEL) được thiết lập.

Bảng sau đây liệt kê mức độ ngắt, một số nguyên từ 0 đến 3 biểu thị mức độ ưu tiên ngắt, cho từng loại ngắt.

Loại ngắt	Mức ngắt
Ngắt mô phỏng (NMICE)*1	3
Ngắt không thể vô hiệu hóa (NMI)	2
Ngắt phần mềm (SWI)	1
Ngắt có thể vô hiệu hóa (MI)	1

*1 Yêu cầu ngắt này yêu cầu một trình mô phỏng trên mạch(ICE). Nó không khả dụng cho các chương trình ứng dụng của người dùng.

Cài đặt mức ngoại lệ (ELEVEL) bằng 0 cho biết không có yêu cầu ngắt nào đang chờ xử lý.

Mức độ ngắt càng cao thì mức độ ưu tiên càng lớn.

Khi phần cứng chấp nhận yêu cầu ngắt, nó sẽ lưu mức ngắt trong trường ELEVEL của PSW.

Khi phần cứng U16 nhận được yêu cầu ngắt, trước tiên nó so sánh mức ngắt với cài đặt mức ngoại lệ hiện tại (ELEVEL). Nếu mức ngắt bằng hoặc lớn hơn ELEVEL, phần cứng sẽ tải bộ đếm chương trình (PC) từ mục nhập thích hợp trong bảng vectơ.

Địa chỉ trong bảng vectơ	Mô tả
0000H	Giá trị ban đầu cho con trỏ ngăn xếp(SP)
0002	Điểm vào quy trình reset cho đầu vào đặt lại bên ngoài hoặc lệnh BRK với cài đặt mức ngoại lệ (ELEVEL) là 2 hoặc 3
0004	Điểm vào quy trình reset cho lệnh BRK với thiết lập mức ngoại lệ (ELEVEL) là 0 hoặc 1
0006H	Điểm vào trình xử lý ngắt cho ngắt giả lập (NMICE)
0008H -	Điểm vào trình xử lý ngắt cho ngắt không thể vô hiệu hóa (NMI) và ngắt có thể vô hiệu hóa (MI)
0080H - 00FEH	Điểm vào trình xử lý ngắt cho ngắt phần mềm (SWI)

Sau đây là quy trình chấp nhận chi tiết cho từng loại ngắt.

1.3.7.2 Ngắt không thể vô hiệu hóa (NMI)

Các chương trình ứng dụng của người dùng không có cách nào để vô hiệu hóa các ngắt NMI. Khi phần cứng phát hiện ra một ngắt, quyền điều khiển sẽ ngay lập tức chuyển đến trình xử lý ngắt NMI thích hợp. Nếu CPU đã thực thi trình xử lý ngắt NMI, quyền điều khiển vẫn quay lại ban đầu.

Tuy nhiên, phần cứng sẽ vô hiệu hóa chúng trong các tình huống sau.

- Giữa một lệnh reset (hoặc đầu vào reset phần cứng hoặc lệnh BRK với cài đặt ELEVEL là 3) và kết thúc lệnh đầu tiên trong trình xử lý reset
- Giữa thời điểm bắt đầu chu kỳ chấp nhận ngắt và thời điểm kết thúc lệnh đầu tiên trong trình xử lý ngắt
- Giữa lệnh tiền tố DSR và lệnh tiếp theo ngay sau đó

Một yêu cầu ngắt không thể vô hiệu hóa sẽ tự động khiến phần cứng thực hiện các hành động sau.

1. Lưu PC trong ELR2.
2. Lưu CSR trong ECSR2.
3. Lưu PSW trong EPSW2.
4. Đặt trường ELEVEL trong PSW thành 2.
5. Reset CSR về không.
6. Tải bộ đếm chương trình (PC) từ bảng vector.
7. Vô hiệu hóa các yêu cầu ngắt trong thời gian diễn ra lệnh đầu tiên trong trình xử lý ngắt.

Thời gian xử lý cần thiết cho các hành động trên là 3 chu kỳ, tuy nhiên, khi sự gián đoạn xảy ra ngay sau lệnh sử dụng địa chỉ [EA+], trình tự gián đoạn được bắt đầu sau khi một chu kỳ máy của các chu kỳ chờ được thực hiện. Để biết thêm chi tiết, hãy xem Phần 3.3 “Thời gian thực thi lệnh.”

Trình xử lý ngắt NMI có thể thoát theo nhiều cách khác nhau. Để biết thêm chi tiết, hãy xem Phần 1.4 “Mức độ ngoại lệ và phòng ghi dự phòng.”

1.3.7.3 Ngắt có thể vô hiệu hóa (MI)

Các ngắt có thể vô hiệu hóa có nhiều nguồn trong số các thiết bị ngoại vi trên bo mạch và các chân đầu vào bên ngoài. Phần cứng chỉ chấp nhận chúng nếu bit MIE trong PSW là “1.”

Tuy nhiên, phần cứng vô hiệu hóa chúng trong các tình huống sau.

- Giữa một lệnh reset (hoặc đầu vào reset phần cứng hoặc lệnh BRK với cài đặt ELEVEL là 3) và kết thúc lệnh đầu tiên trong trình xử lý reset
- Giữa thời điểm bắt đầu chu kỳ chấp nhận ngắt và thời điểm kết thúc lệnh đầu tiên trong trình xử lý ngắt
- Giữa lệnh tiền tố DSR và lệnh tiếp theo ngay sau đó
- Khi cài đặt ELEVEL là 2 hoặc 3

Việc chấp nhận yêu cầu ngắt có thể vô hiệu hóa sẽ tự động khiến phần cứng thực hiện các hành động sau.

1. Lưu PC trong ELR1.
2. Lưu CSR trong ECSR1.
3. Lưu PSW trong EPSW1.
4. Đặt trường ELEVEL trong PSW thành 1.
5. Đặt bit MIE trong PSW thành “0” để vô hiệu hóa các yêu cầu ngắt tiếp theo.
6. Reset CSR về không.
7. Tải bộ đếm chương trình (PC) từ bảng vector.
8. Vô hiệu hóa tất cả các yêu cầu ngắt trong thời gian diễn ra lệnh đầu tiên trong trình xử lý ngắt.

Thời gian xử lý cần thiết cho các hành động trên là 3 chu kỳ, tuy nhiên, khi sự gián đoạn xảy ra ngay sau lệnh sử dụng địa chỉ [EA+], trình tự gián đoạn được bắt đầu sau khi một chu kỳ máy của các chu kỳ chờ được thực hiện. Để biết thêm chi tiết, hãy xem Phần 3.3 “Thời gian thực thi lệnh.”

Trình xử lý ngắt MI có thể thoát theo nhiều cách khác nhau. Để biết thêm chi tiết, hãy xem Phần 1.4 “Mức độ ngoại lệ và phòng ghi dự phòng.”

1.3.7.4 Ngắt phần mềm (SWI)

Các ngắt phần mềm đến từ bên trong chương trình ứng dụng của người dùng, do đó được chấp nhận ngay lập tức. Toán hạng của lệnh SWI chỉ định số ngắt.

Yêu cầu ngắt phần mềm sẽ tự động khiến phần cứng thực hiện các hành động sau.

1. Lưu PC trong ELR1.
2. Lưu CSR trong ECSR1.
3. Lưu PSW trong EPSW1.
4. Đặt trường ELEVEL trong PSW thành 1.
5. Đặt bit MIE trong PSW thành “0” để vô hiệu hóa các yêu cầu ngắt tiếp theo.
6. Reset CSR về không.
7. Tải bộ đếm chương trình (PC) từ bảng vector.
8. Vô hiệu hóa tất cả các yêu cầu ngắt trong thời gian diễn ra lệnh đầu tiên trong trình xử lý ngắt.

Thời gian xử lý cần thiết cho các hành động trên là 3 chu kỳ, tuy nhiên, khi lệnh SWI được thực hiện ngay sau lệnh sử dụng địa chỉ [EA+], trình tự ngắt được bắt đầu sau khi thực hiện một chu kỳ máy của các chu kỳ chờ. Để biết thêm chi tiết, hãy xem Phần 3.3 “Thời gian thực thi lệnh.”

Trình xử lý ngắt phần mềm có thể thoát theo nhiều cách khác nhau. Để biết thêm chi tiết, hãy xem Phần 1.4 “Mức độ ngoại lệ và phòng ghi dự phòng.”

1.4 Mức độ ngoại lệ và Thanh ghi dự phòng

Kiến trúc U16 cung cấp ba bộ thanh ghi sao lưu để lưu nội dung của bộ đếm chương trình (PC), thanh ghi phân đoạn mã (CSR) và thanh ghi trạng thái chương trình (PSW) trong khi gọi chương trình con và trình xử lý ngắt. Thanh ghi sao lưu PC và CSR áp dụng cho cả hai tình huống; thanh ghi PSW chỉ áp dụng cho trình xử lý ngắt.

Bảng sau đây tóm tắt việc sử dụng các thanh ghi sao lưu này.

Thanh ghi sao lưu PC

Tên	Mô tả
LR	Phần này giữ offset của địa chỉ trả về cho lệnh gọi chương trình con với lệnh BL.
ELR1	Phần này giữ offset của địa chỉ trả về cho một ngắt có thể vô hiệu hóa hoặc một lệnh SWI.
ELR2	Phần này giữ offset của địa chỉ trả về cho một ngắt không thể vô hiệu hóa.
ELR3	Phần này giữ offset của địa chỉ trả về cho một ngắt mô phỏng.

Thanh ghi sao lưu CSR

Tên	Mô tả
LCSR	Phần này giữ phần phân đoạn vật lý của địa chỉ trả về cho lệnh gọi chương trình con với lệnh BL.
ECSR1	Phần này giữ phần phân đoạn vật lý của địa chỉ trả về cho một ngắt có thể vô hiệu hóa hoặc một lệnh SWI.
ECSR2	Phần này giữ phần phân đoạn vật lý của địa chỉ trả về cho một ngắt không thể vô hiệu hóa.
ECSR3	Phần này giữ phần phân đoạn vật lý của địa chỉ trả về cho một ngắt mô phỏng.

Thanh ghi sao lưu trạng thái chương trình (PSW)

Tên	Mô tả
EPSW1	Phần này giữ PSW ngay trước khi có ngắt có thể vô hiệu hóa hoặc lệnh SWI.
EPSW2	Phần này giữ PSW ngay trước khi có ngắt không thể vô hiệu hóa.
EPSW3	Phần này giữ PSW ngay trước khi có ngắt mô phỏng.

LR và LCSR được lưu bằng lệnh BL gọi một chương trình con và được khôi phục bằng lệnh RT kết thúc chương trình con.

Các thanh ghi ELR, ECSR và EPSW được sử dụng phụ thuộc vào giá trị chỉ mục từ trường ELEVEL trong thanh ghi trạng thái chương trình (PSW). Phần cứng lưu vào chúng trong chu kỳ chấp nhận ngắt; lệnh RTI ở cuối trình xử lý ngắt khôi phục từ chúng.

Lưu ý rằng các bộ chỉ cung cấp một thanh ghi cho mỗi cấp độ. Do đó, nếu các chương trình con hoặc trình xử lý ngắt được lồng nhau, thì không thể trả về bằng các lệnh RT và RTI thường được sử dụng. Chương trình con hoặc trình xử lý ngắt phải sử dụng các lệnh PUSH để lưu nội dung thanh ghi vào ngăn xếp trước khi lồng nhau và trả về bằng các lệnh POP thay thế.

Việc lựa chọn phương pháp thích hợp để lưu các thanh ghi này phụ thuộc vào trạng thái CPU, do đó cần đặc biệt chú ý trong giai đoạn thiết kế. Các trang sau đây cho thấy cách điều chỉnh chương trình ứng dụng của người dùng theo trạng thái thực thi của U16.

Sau đây mô tả các cân nhắc về lập trình cho mỗi thiết lập ELEVEL có thể có và liệu chương trình ứng dụng của người dùng có lồng ghép chương trình con và ngắt hay không.

A: Bất kỳ ngắt nào không được xử lý

Trạng thái chạy của mức ngoại lệ (ELEVEL) là 0 và các thanh ghi sao lưu được sử dụng là LR và LCSR. Cách các thủ tục bắt đầu và kết thúc chỉ phụ thuộc vào việc các chương trình con có được lồng nhau hay không.

A-1: Khi một chương trình con không được gọi bởi chương trình trong một chương trình con

- Xử lý ngay sau khi bắt đầu chương trình con

Không có ghi chú cụ thể.

- Xử lý ở cuối chương trình con

Khôi phục PC từ LR bằng lệnh RT.

Ví dụ mô tả: State A-1

```
Sub_A-1:        ; bắt đầu chương trình
:
:               ; Khôi phục PC từ LR.
                ; Kết thúc chương trình con
```

A-2: Khi một chương trình con được gọi bởi chương trình trong một chương trình con.

- Xử lý ngay sau khi bắt đầu chương trình con

Sử dụng lệnh PUSH LR để lưu địa chỉ trả về vào ngăn xếp.

- Xử lý ở cuối chương trình con

Sử dụng lệnh POP PC thay vì RT để trả về từ chương trình con.

Ví dụ mô tả: State A-2

```
Sub_A-2:        ; Bắt đầu chương trình con.
  PUSH  LR      ; Lưu địa chỉ trả về vào ngăn xếp
  :
  :
  BL    Sub_1   ; Gọi chương trình con lồng nhau Sub_1
  :
  POP   PC      ; Khôi phục PC từ ngăn xếp
                ; Kết thúc chương trình con
```

```
Sub_1:  ;Bắt đầu chương trình con
  :
  :
  RT    ; Khôi phục PC từ LR.
        ; Kết thúc chương trình con
```

B: Ngắt có thể vô hiệu hóa đang được xử lý

Trạng thái chạy của mức ngoại lệ (ELEVEL) là 1 và các thanh ghi sao lưu được sử dụng là ELR1, ECSR1 và EPSW1. Cách các thủ tục bắt đầu và kết thúc phụ thuộc vào việc nhiều ngắt được bật hay tắt.

B-1:Khi một chương trình con không được gọi bởi chương trình khi thực hiện một quá trình ngắt.

B-1-1: Khi nhiều ngắt bị vô hiệu hóa.

- Xử lý ngay sau khi bắt đầu thực hiện quá trình ngắt

  Không có ghi chú cụ thể.

- Xử lý khi kết thúc quá trình thực hiện ngắt

  Khôi phục PC từ ELR1 và PSW từ EPSW1 bằng lệnh RTI.

Ví dụ mô tả: State B-1-1

```
Intrpt_B-1-1:          ; Bắt dầu của quá trình ngắt.
  :
  :                    ; Trả về PC từ ELR
                       ; Trả về PSW từ EPSW
                       ; Kết thúc
```

B-1-2 : Khi nhiều ngắt được kích hoạt.

- Xử lý ngay sau khi bắt đầu thực hiện quá trình ngắt

  Chỉ định “PUSH ELR, EPSW” để lưu địa chỉ trả về ngắt và trạng thái PSW trong ngăn xếp.

- Xử lý khi kết thúc thực hiện quá trình ngắt

  Chỉ định “POP PSW, PC” thay vì lệnh RTI để trả lại nội dung của ngăn xếp cho PC và PSW.

Example of description: State B-1-2

```
Intrpt_B-1-2:          ; Bắt đầu
   PUSH   ELR, EPSW    ; Lưu ELR và EPSW tại điểm bắt
   :
   EI                  ; Kích hoạt ngắt
   :
   :
   POP PSW,PC          ; Trả về PC từ ngăn xếp
                       ; Trả về PSW từ ngăn xếp
                       ; Kết thúc
```

B-2: Khi một chương trình con được gọi bởi chương trình trong khi đang thực hiện một quá trình ngắt.

B-2-1: Khi nhiều ngắt bị vô hiệu hóa.

- Xử lý ngay sau khi bắt đầu thực hiện quá trình ngắt

Chỉ định lệnh “PUSH LR” để lưu địa chỉ trả về của chương trình con trong ngăn xếp.

- Xử lý khi kết thúc thực hiện quá trình ngắt

Chỉ định “POP LR” ngay trước lệnh RTI để trả về từ quá trình xử lý ngắt sau khi trả về địa chỉ trả về của chương trình con cho LR.

Example of description: State B-2-1

```
Intrpt_B-2-1:    ; Bắt đầu
  PUSH  LR       ; Lưu địa chỉ trả về tại điểm bắt đầu
  :
  :
  BL Sub_1       ; Gọi chương trình con lồng nhau Sub_1
  :
  POP  LR        ; Khôi phục LR từ ngăn xếp
  RTI            ; Kết thúc
```

```
Sub_1:    ;Điểm bắt đầu của chương trình con
 :
 :
RT        ; Trả về PC từ LR
          ; Kết thúc của chương trình con
```

B-2-2 : Khi nhiều ngắt được kích hoạt.

- Xử lý ngay sau khi bắt đầu thực hiện quá trình ngắt

Chỉ định “PUSH ELR, EPSW, LR” để lưu địa chỉ trả về ngắt, địa chỉ trả về chương trình con và trạng thái EPSW trong ngăn xếp.

- Xử lý khi kết thúc quá trình thực hiện ngắt

Chỉ định “POP PC, PSW, LR” thay cho lệnh RTI để trả về dữ liệu đã lưu của địa chỉ trả về ngắt cho PC, dữ liệu đã lưu của EPSW cho PSW và dữ liệu đã lưu của LR cho LR.

Ví dụ mô tả: State B-2-2

```
Intrpt_B-2-2:              ; Bắt đầu
  PUSH   ELR,EPSW,LR       ; Lưu ELR, EPSW, LR tại điểm bắt đầu
  :
  EI                       ; Kích hoạt ngắt
  :
  BL Sub_1                 ; Gọi chương trình con Sub_1
  :
  POP   PC,PSW,LR          ; Trả về PC từ ngăn xếp
                           ; Trả về PSW từ ngăn xếp
                           ; Trả về LR từ ngăn xếp
                           ; Kết thúc
```

```
Sub_1:    ; Điểm bắt đầu của chương trình con
:
:
RT        ; Trả về PC từ LR
          ; Kết thúc của chương trình con
```

C: Ngắt không thể vô hiệu hóa đang được xử lý

Trạng thái chạy của mức ngoại lệ (ELEVEL) là 2 và các thanh ghi sao lưu được sử dụng là ELR2, ECSR2 và EPSW2. Cách các thủ tục bắt đầu và kết thúc phụ thuộc vào việc có gọi chương trình con hay không.

C-1: Khi một chương trình con không được chương trình gọi khi thực hiện một quá trình ngắt.

- Xử lý ngay sau khi bắt đầu thực hiện quá trình ngắt

Chỉ định “PUSH ELR, EPSW” để lưu địa chỉ trả về ngắt, địa chỉ trả về chương trình con và trạng thái EPSW trong ngăn xếp.

- Xử lý khi kết thúc thực hiện quá trình ngắt

Chỉ định “POP PSW, PC” thay cho lệnh RTI để trả về dữ liệu đã lưu của địa chỉ trả về ngắt cho PC, dữ liệu đã lưu của EPSW cho PSW.

Example of description: State C-1

```
Intrpt_C-1:              ; Bắt đầu
  PUSH   ELR,EPSW        ; Lưu ELR và EPSW tại điểm bắt
  :
  :
  POP  PSW,PC            ; Trả về PC từ ngăn xếp
                         ; Trả về PSW từ ngăn xếp
                         ; Kết thúc
```

C-2: Khi một chương trình con được gọi bởi chương trình trong khi thực thi một quá trình ngắt.

- Xử lý ngay sau khi bắt đầu thực thi quá trình ngắt

Chỉ định “PUSH ELR, EPSW, LR” để lưu địa chỉ trả về ngắt, địa chỉ trả về của chương trình con và trạng thái EPSW trong ngăn xếp.

- Xử lý khi kết thúc quá trình thực thi ngắt

Chỉ định “POP PSW, PC, LR” thay cho lệnh RTI để trả về dữ liệu đã lưu của địa chỉ trả về ngắt cho PC, dữ liệu đã lưu của EPSW cho PSW và dữ liệu đã lưu của LR cho LR.

Ví dụ mô tả: State

```
C-2 Intrpt_C-2:          ; Bắt đầu
  PUSH   ELR,EPSW,LR     ; Lưu ELR, EPSW, LR ở
                             vị trí ban đầu
  :
  :
  BL Sub_1               ; Gọi chương trình con
  :
  POP   PSW,PC,LR        ; Trả về PC từ ngăn xếp
                         ; Trả về PSW từ ngăn xếp
                         ; Trả về LR từ ngăn xếp
                         ; Kết thúc
```

```
Sub_1        ; Bắt đầu
:
:
:            ; Trả về PC từ LR
             ; Kết thúc của chương
             trình con
```

Việc lựa chọn phương pháp thích hợp để lưu các thanh ghi này phụ thuộc vào trạng thái CPU, do đó cần đặc biệt chú ý trong giai đoạn thiết kế..

1.5 Ghi chú về ngắt không thể vô hiệu hóa

Mục này mô tả các ghi chú về ngắt không thể vô hiệu hóa tại thời điểm phát triển chương trình của C.

Vì yêu cầu ngắt không thể vô hiệu hóa không thể bị vô hiệu hóa, nên nó cần xử lý ngắt nhiều lần đối với ngắt không thể vô hiệu hóa trong chương trình ứng dụng. Nếu biện pháp chống lại ngắt nhiều lần không được triển khai, chương trình có thể không thể trả về từ một quá trình ngắt.

Do đó, khi các chip mục tiêu sử dụng các ngắt không thể vô hiệu hóa, vui lòng đặt danh mục là hai theo một pragma INTERRUPT.

```
  static void int_WDTINT(void);
  #pragma interrupt int_WDTINT
  0x08 2 static void
  int_WDTINT(void)
{
    :
    :
}
```

Mã lắp ráp mà trình biên dịch đưa ra cho chương trình C được đề cập ở trên là như sau.

```
  _int_WDTINT :
    push elr, epsw
    :
    :
    pop psw, pc
```

* Khi một chương trình ngắt chứa lệnh gọi hàm, ngoài nhóm thanh ghi sao lưu, mã PUSH/POP của nhóm thanh ghi sao lưu đã đề cập ở trên và nhóm thanh ghi (LR, EA, R0-R3) có thể được thay đổi trong hàm được gọi sẽ được tạo ra.

1.6 Chặn ngắt

Như đã đề cập ở trên, phần cứng vô hiệu hóa tất cả các yêu cầu ngắt đang chờ xử lý trong suốt thời gian của các tình huống sau.

(1) Giữa thời điểm bắt đầu chu kỳ chấp nhận ngắt và thời điểm kết thúc lệnh đầu tiên trong trình xử lý ngắt

Phần cứng trì hoãn việc chấp nhận yêu cầu ngắt mới cho đến khi nó hoàn tất việc thực thi lệnh đầu tiên trong trình xử lý ngắt cho lệnh cũ.

(2) Giữa lệnh tiền tố DSR và lệnh tiếp theo ngay sau đó

Phần cứng trì hoãn việc chấp nhận yêu cầu ngắt mới cho đến khi nó hoàn tất việc thực thi cả hai lệnh trong cặp.

Để biết thêm chi tiết về các lệnh tiền tố DSR, hãy xem Phần 1.3.4 “Các lệnh tiền tố DSR”. Mặc dù một chuỗi các lệnh tiền tố DSR kích hoạt lại đúng các yêu cầu ngắt, nhưng các chuỗi như vậy không được xuất hiện trong mã chương trình vì chúng có thể dẫn đến hành vi không mong muốn.

1.7 Sửa đổi ngăn xếp

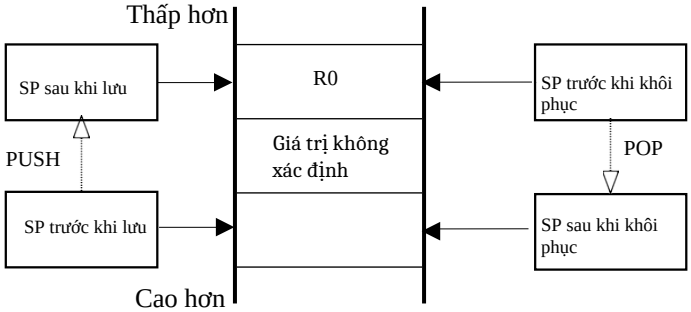
Phần này tóm tắt các tác động mà lệnh PUSH và POP có trên ngăn xếp. Để biết thêm chi tiết, hãy xem Chương 3 “Mô tả Lệnh.”

Con trỏ ngăn xếp (SP) luôn di chuyển theo một số byte chẵn. Nếu toán hạng lệnh PUSH biểu diễn một số byte lẻ, phần cứng sẽ lưu trữ ngữ cảnh của toán hạng đó bằng một byte giả. Tương tự, nếu toán hạng lệnh POP biểu diễn một số byte lẻ, phần cứng sẽ khôi phục thanh ghi và đọc một byte giả mà không khôi phục bất kỳ thanh ghi nào.

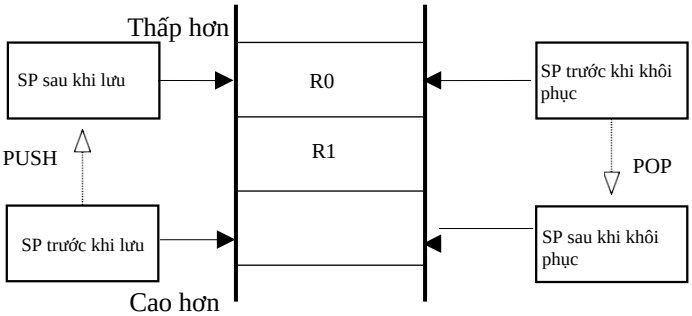
Lưu ý rằng hoạt động của các lệnh chỉ định LR hoặc ELR làm toán hạng phụ thuộc vào mô hình bộ nhớ phần cứng.

Các hình sau đây minh họa hoạt động của hai lệnh này.

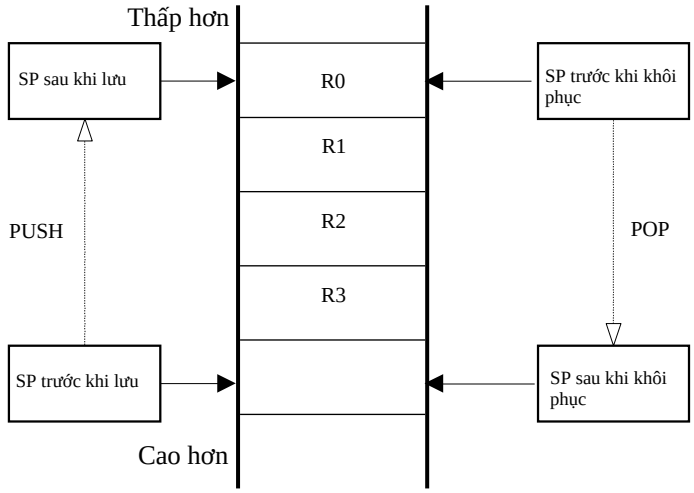
PUSH R0 / POP R0



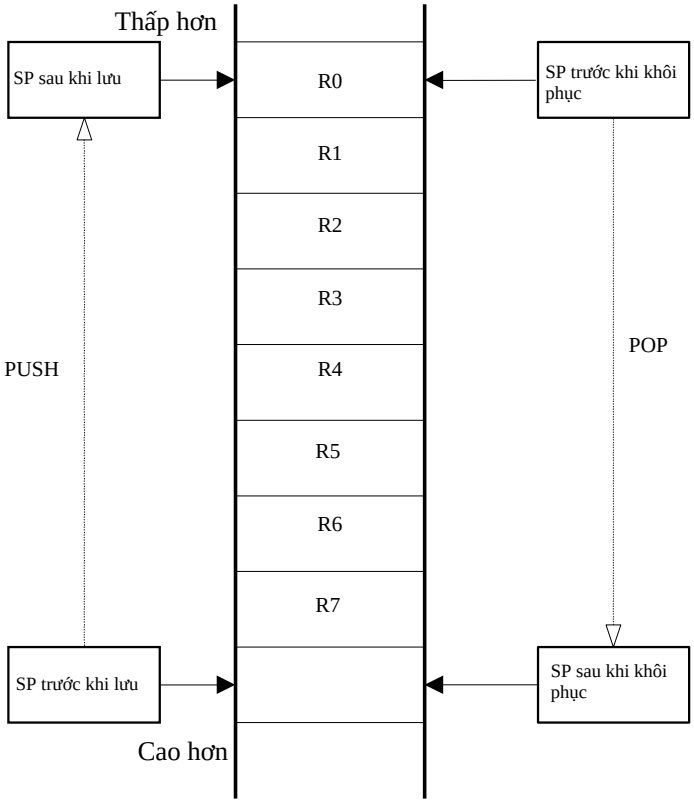
PUSH ER0 / POP ER0



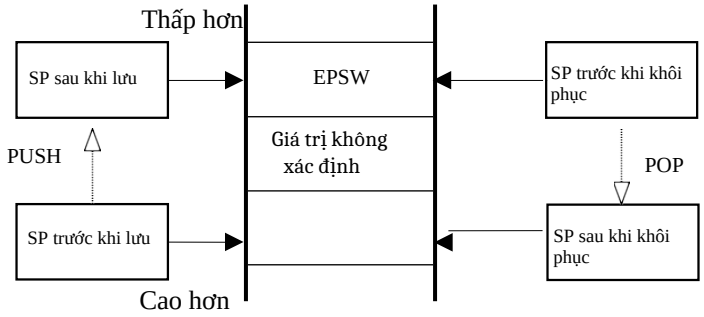
PUSH XR0 / POP XR0



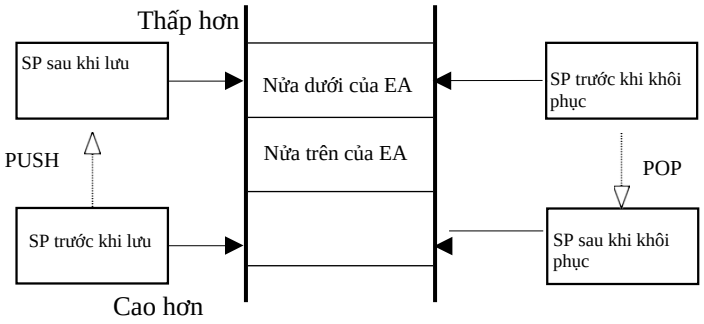
PUSH QR0 / POP QR0



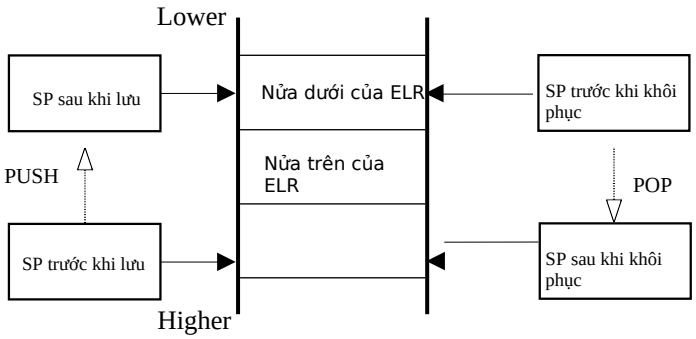
PUSH EPSW / POP PSW



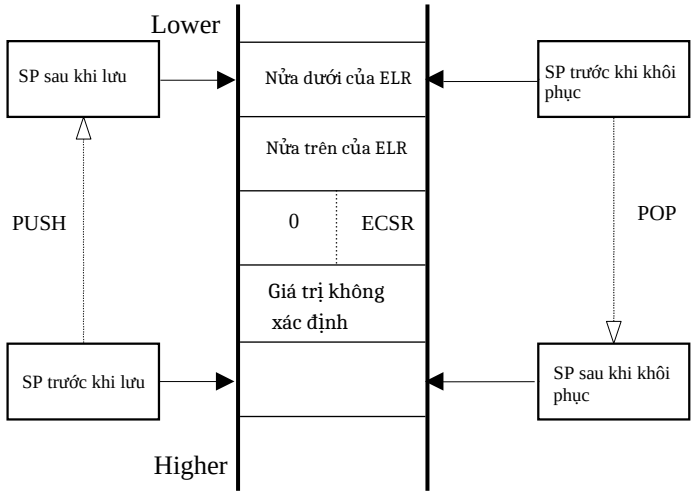
PUSH EA / POP EA



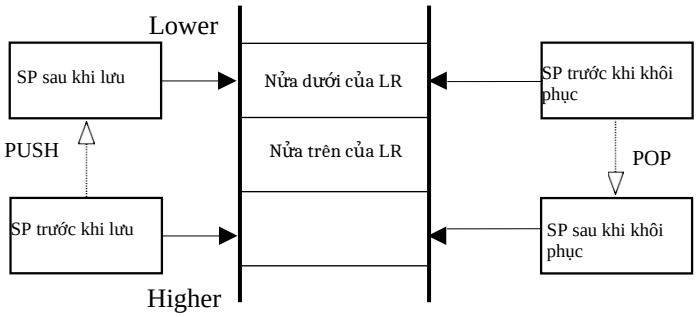
PUSH ELR / POP PC (SMALL model)



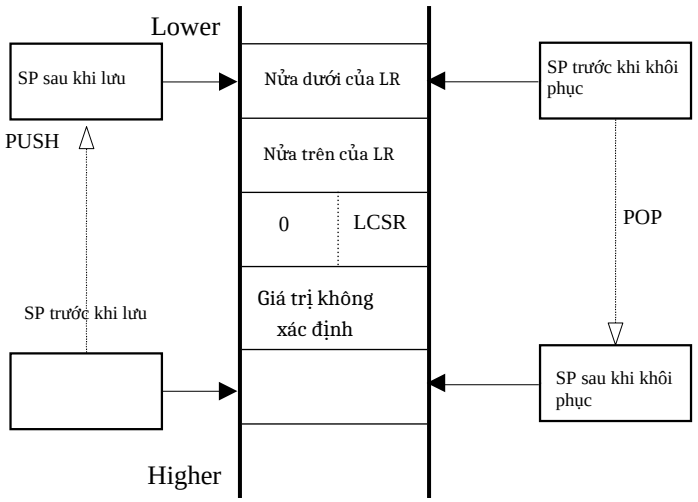
PUSH ELR / POP PC (LARGE model)



PUSH LR / POP LR (SMALL model)



PUSH LR / POP LR (LARGE model)



2. Các loại địa chỉ

2.1 Các loại địa chỉ

Kiến trúc nX-U16/100 có bốn loại địa chỉ:

- Địa chỉ thanh ghi để truy cập các thanh ghi nội bộ và thanh ghi đồng xử lý
- Địa chỉ bộ nhớ để truy cập bộ nhớ dữ liệu và bộ nhớ chương trình/mã bên trong cửa sổ ROM
- Địa chỉ tức thời để chỉ định các giá trị số
- Địa chỉ bộ nhớ chương trình/mã để truy cập bộ nhớ chương trình/mã

2.2 Địa chỉ thanh ghi

Các loại địa chỉ thanh ghi sau đây truy cập vào nội dung của thanh ghi được chỉ định.

Ký hiệu địa chỉ	Chức năng
R*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi chung có kích thước byte được chỉ định (Rn).
ER*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi chung có kích thước word được chỉ định (ERn). Khi bảng hướng dẫn liệt kê ERn trong một toán hạng, BP có thể được thay thế cho ER12 và FP cho ER14.
XR*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi chung có kích thước double word được chỉ định (XRn).
QR*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi chung có kích thước quad word (QRn) được chỉ định.
CR*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi đồng xử lý có kích thước byte được chỉ định (CRn).
CER*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi đồng xử lý có kích thước word được chỉ định (CERn).
CXR*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi đồng xử lý có kích thước double word được chỉ định (CXRn).
CQR*n*	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi đồng xử lý có kích thước quad word (CQRn) được chỉ định.
PC	Kiểu địa chỉ này truy cập vào nội dung của bộ đếm chương trình (PC).
LR	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi liên kết (LR).
EA	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi EA.
SP	Kiểu địa chỉ này truy cập vào nội dung của con trỏ ngăn xếp.
PSW	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi trạng thái chương trình (PSW).
ELR	Kiểu địa chỉ này truy cập vào nội dung của một thanh ghi liên kết ngoại lệ (ELR).
ECSR	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi sao lưu CSR.
EPSW	Kiểu địa chỉ này truy cập vào nội dung của thanh ghi sao lưu PSW.
R*n.bit_offset*	Kiểu địa chỉ này truy cập vào nội dung của bit được chỉ định bởi bit_offset trong thanh ghi chung Rn.

2.3 Địa chỉ bộ nhớ

Kiểu định danh địa chỉ này truy cập nội dung của một địa chỉ trong không gian bộ nhớ dữ liệu.

Truy cập dữ liệu trong một phân đoạn vật lý khác với phân đoạn vật lý #0 đòi hỏi phải thao tác với thanh ghi phân đoạn dữ liệu (DSR) bằng lệnh tiền tố DSR.

Để ngăn chặn hoạt động không mong muốn và cung cấp khả năng kiểm tra mạnh nhất có thể đối với quyền truy cập bộ nhớ, các thông số kỹ thuật ngôn ngữ lắp ráp U16 cố tình cấm sử dụng các lệnh tiền tố DSR trong mã nguồn chương trình. Thay vào đó, hãy sử dụng tiền tố DSR tương ứng bên trong chính lệnh truy cập bộ nhớ..

Lệnh tiền tố DSR	Chức năng	Tiền tố tương ứng
`1110_0011_iiii_iiii`	Tải DSR với giá trị tức thời 8 bit iiii_iiii.	*pseg_addr* : hoặc FAR
`1001_0000_dddd_1111`	Tải DSR với nội dung của thanh ghi chung Rd.	R*d* :
`1111_1110_1001_1111`	Sử dụng giá trị DSR hiện tại.	DSR :

Bảng sau đây hiển thị các ví dụ về các tiền tố này ở bên trái và kết quả của chúng ở bên phải.

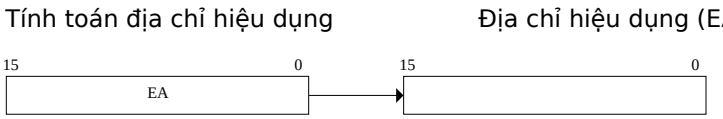
Assembly Language Source Code	Trình tự lệnh thực tế
`L R0, 1:2345H`	`DSR ← 1` `R0 ← [2345H]`
`L R0, R1:[ER2]`	`DSR ← R1` `R0 ← [ER2]`
`ST R1, DSR:[EA]`	`[(DSR<<16)|EA] ← R1`

Các phần gạch chân trong Bảng trên chỉ ra các tiền tố DSR tạo ra các thao tác DSR mong muốn.

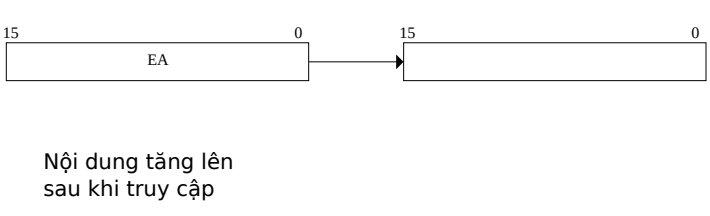
Nếu không có tiền tố DSR, lệnh sẽ truy cập phân đoạn vật lý #0 trong không gian bộ nhớ dữ liệu.

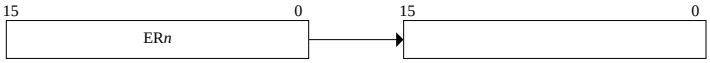
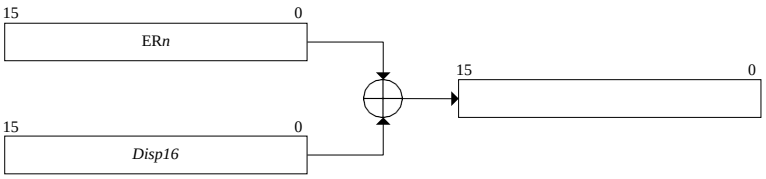
2.3.1 Thanh ghi Địa chỉ gián tiếp

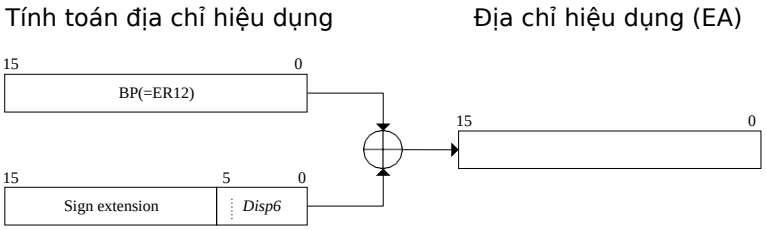
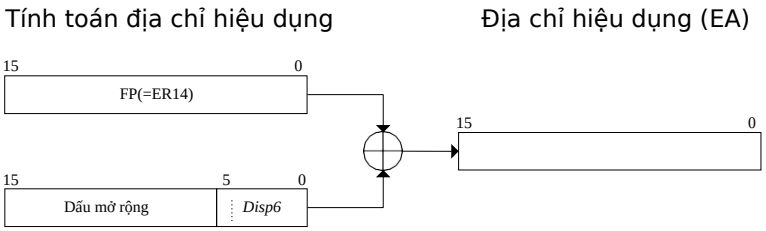
Các loại thanh ghi địa chỉ gián tiếp sau đây truy cập nội dung của địa chỉ bộ nhớ dữ liệu trong thanh ghi được chỉ định.

Ký hiệu địa chỉ	Chức năng
[EA]	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại vị trí offset trong thanh ghi EA. 
pseg_addr:[EA]	Biến thể này sử dụng số phân đoạn vật lý được chỉ định bởi *#pseg_addr*.
DSR:[EA]	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:[EA]	Biến thể này sử dụng số phân đoạn vật lý trong thanh ghi chung R*d*.
[EA+]	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại vị trí offset trong thanh ghi EA. Sau khi truy cập, nội dung của thanh ghi EA được tăng lên theo kích thước toán hạng tính bằng byte và đối với tất cả các kích thước ngoại trừ byte, được làm tròn xuống thành một địa chỉ chẵn.

Kích thước toán hạng	Nội dung EA	Tăng
Byte	Chẵn	1
	Lẻ	1
Word	Chẵn	2
	Lẻ	1
Double word	Chẵn	4
	Lẻ	3
Quad word	Chẵn	8
	Lẻ	7

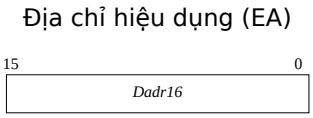
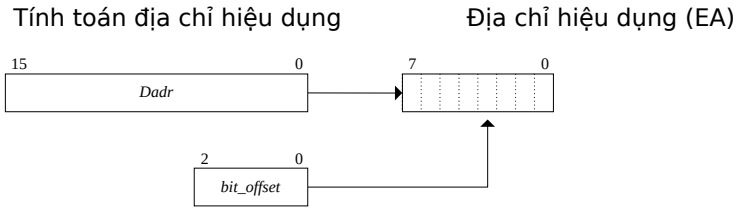


Ký hiệu địa chỉ	Chức năng
pseg_addr:[EA+]	Biến thể này sử dụng số phân đoạn vật lý được chỉ định bởi #*pseg_addr*.
DSR:[EA+]	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:[EA+]	Biến thể này sử dụng số phân đoạn vật lý trong thanh ghi chung R*d*.
[ER*n*]	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại vị trí offset trong thanh ghi chung có kích thước word ER*n*. [BP] thay vì [ER12] và [FP] thay vì [ER14] cũng được chấp nhận. Tính toán địa chỉ hiệu dụng Địa chỉ hiệu dụng (EA) 
pseg_addr:[ER*n*]	Biến thể này sử dụng số phân đoạn vật lý được chỉ định bởi #*pseg_addr*.
DSR:[ER*n*]	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:[ER*n*]	Biến thể này sử dụng số phân đoạn vật lý trong thanh ghi chung R*d*.
Disp16[ER*n*]	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại địa chỉ byte được hình thành bằng cách thêm độ dịch chuyển *Disp16* vào nội dung của thanh ghi chung có kích thước word ER*n*. Tính toán địa chỉ hiệu dụng Địa chỉ hiệu dụng (EA) 
pseg_addr:*Disp16*[ER*n*]	Biến thể này sử dụng số phân đoạn vật lý được chỉ định bởi #*pseg_addr*.
DSR:*Disp16*[ER*n*]	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:*Disp16*[ER*n*]	Biến thể này sử dụng số phân đoạn vật lý trong thanh ghi chung R*d*.

Ký hiệu địa chỉ	Chức năng
Disp6[BP]	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại địa chỉ byte được hình thành bằng cách thêm độ dịch chuyển mở rộng dấu *Disp6* vào nội dung của con trỏ cơ sở (BP). Nếu không có tiền tố DSR, kiểu địa chỉ này sẽ truy cập phân đoạn vật lý #0 trong không gian bộ nhớ dữ liệu. 
pseg_addr:*Disp6*[BP]	Biến thể này sử dụng số phân đoạn vật lý được chỉ định bởi #*pseg_addr*.
DSR:*Disp6*[BP]	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:*Disp6*[BP]	Biến thể này sử dụng số phân đoạn vật lý trong số đăng ký chung R*d*.
Disp6[FP]	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại địa chỉ byte được hình thành bằng cách thêm độ dịch chuyển mở rộng dấu *Disp6* vào nội dung của con trỏ khung (FP). Nếu không có tiền tố DSR, kiểu địa chỉ này sẽ truy cập phân đoạn vật lý #0 trong không gian bộ nhớ dữ liệu. 
pseg_addr:*Disp6*[FP]	Biến thể này sử dụng số phân đoạn vật lý được chỉ định bởi #*pseg_addr*.
DSR:*Disp6*[FP]	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:*Disp6*[FP]	Biến thể này sử dụng số phân đoạn vật lý trong thanh ghi chung R*d*.

2.3.2 Địa chỉ trực tiếp

Các kiểu định danh địa chỉ trực tiếp sau đây truy cập nội dung của địa chỉ bộ nhớ dữ liệu được chỉ định.

Ký hiệu địa chỉ	Chức năng
Dadr	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại địa chỉ byte trong lệnh.. 
pseg_addr:*Dadr*	Biến thể này sử dụng số phân đoạn vật lý *pseg_addr*.
DSR:*Dadr*	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:*Dadr*	Biến thể này sử dụng số phân đoạn vật lý trong thanh ghi chung R*d*.
Dbitadr	Kiểu địa chỉ này truy cập vào nội dung của không gian bộ nhớ dữ liệu tại địa chỉ bit (*Dadr.bit_offset*) trong lệnh. 
pseg_addr:*Dbitadr*	Biến thể này sử dụng số phân đoạn vật lý *pseg_addr*.
DSR:*Dbitadr*	Biến thể này sử dụng số phân đoạn vật lý trong DSR.
R*d*:*Dbitadr*	Biến thể này sử dụng số phân đoạn vật lý trong thanh ghi chung R*d*.

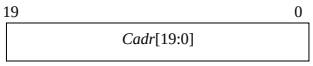
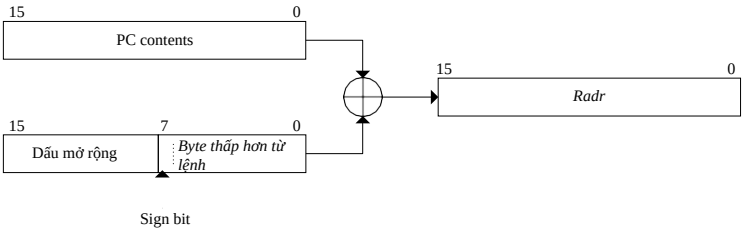
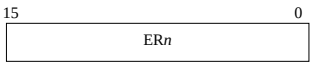
2.4 Địa chỉ tức thì

Các kiểu địa chỉ giá trị tức thời sau đây sử dụng giá trị tức thời có trong lệnh.

Ký hiệu địa chỉ	Chức năng
#imm8	Giá trị đã chỉ định được coi là giá trị tức thời 8 bit.
#signed8	Giá trị đã chỉ định được coi là giá trị tức thời 8 bit có dấu. Lệnh ADD SP, *#imm8* coi *imm8* là *signed8*. Phạm vi hợp lệ cho *signed8* là từ -128 đến +127.
#unsigned8	Giá trị đã chỉ định được coi là giá trị tức thời 8 bit không dấu. Lệnh MOV PSW, *#imm8* coi *imm8* là *unsigned8*. Phạm vi hợp lệ cho *unsigned8* là từ 0 đến 0FFH.
#width	Giá trị được chỉ định được coi là kích thước thay đổi. Phạm vi hợp lệ cho *width* là từ 0 đến 7.
#snum	Giá trị đã chỉ định được coi là số vectơ lệnh SWI. Phạm vi hợp lệ cho *snum7* là từ 0 đến 63.
#imm7	Giá trị đã chỉ định được coi là giá trị tức thời 7 bit có dấu. Phạm vi hợp lệ cho *imm7* là từ -64 đến +63.

2.5 Địa chỉ bộ nhớ chương trình/mã

Các loại địa chỉ sau đây truy cập nội dung của địa chỉ bộ nhớ chương trình/mã.

Ký hiệu địa chỉ	Chức năng
Cadr	Kiểu địa chỉ này chỉ định địa chỉ đích nhánh 20 bit cho lệnh B và BL. Lưu ý rằng nó bao gồm số phân đoạn vật lý, do đó lệnh có thể tạo một nhánh đến một phân đoạn vật lý khác. Địa chỉ hiệu dụng (EA) 
Radr	Kiểu địa chỉ này chỉ định một địa chỉ đích nhánh tương đối cho các lệnh nhánh có điều kiện và các lệnh nhánh được tối ưu hóa. Đích phải nằm trong cùng phân đoạn vật lý. Tính toán địa chỉ hiệu dụng Địa chỉ hiệu dụng (EA) 
ER*n*	Kiểu địa chỉ này chỉ định nội dung của thanh ghi chung có kích thước word ER*n* làm offset mục tiêu nhánh cho lệnh B và BL. Đích phải nằm trong cùng phân đoạn vật lý. Địa chỉ hiệu dụng (EA) 

3. Mô tả lệnh

Chương này mô tả hoạt động chi tiết của từng lệnh.

3.1 Tổng quan

Các lệnh lõi nX-U16/100 có từ 0 đến 2 toán hạng. Khi có 2 toán hạng, toán hạng đầu tiên là đích; toán hạng thứ hai là nguồn.

Các toán hạng này sử dụng các kiểu địa chỉ được mô tả trong Chương 2.

Để dễ giải thích, tài liệu này sử dụng các ký hiệu sau để mô tả hoạt động lệnh.

Kí hiệu	Ý nghĩa
←	Phép gán
+ or $\oplus$	Phép cộng
–	Phép trừ
*	Phép nhân
/	Phép chia
>>	Dịch chuyển sang phải
<<	Dịch chuyển sang trái
=	Bằng
!=	Khác
&	Bitwise AND
|	Bitwise OR
^	Bitwise loại trừ OR
~	Bitwise đảo ngược

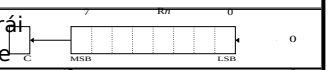
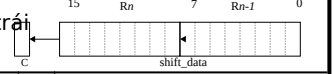
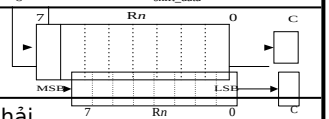
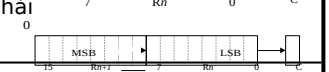
3.2 Lệnh theo Nhóm chức năng

Vui lòng tham khảo Mục 3.4 “Mô tả lệnh” để biết cách vận hành chi tiết của từng lệnh.

Lệnh thuật toán

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
ADD	R*n*	R*m*	*	*	*	*		*	Phép cộng (8 bit)	R*n*← R*n*+obj
		#imm8	*	*	*	*		*		
MOV	R*n*	R*m*		*	*				Truyền dữ liệu (8-bit)	R*n* ← obj
		#imm8		*	*					
ADDC	R*n*	R*m*	*	*	*	*		*	Cộng với carry	R*n*←R*n*+obj+c
		#imm8	*	*	*	*		*		
CMP	R*n*	R*m*	*	*	*	*		*	So sánh (8-bit)	R*n*–obj
		#imm8	*	*	*	*		*		
CMPC	R*n*	R*m*	*	*	*	*		*	So sánh với carry	R*n*–obj–c
		#imm8	*	*	*	*		*		
AND	R*n*	R*m*		*	*				Bitwise AND	R*n*←R*n*&obj
		#imm8		*	*					
OR	R*n*	R*m*		*	*				Bitwise OR	R*n*←R*n* | obj
		#imm8		*	*					
XOR	R*n*	R*m*		*	*				Bitwise loại trừ OR	R*n*←R*n*^obj
		#imm8		*	*					
SUB	R*n*	R*m*	*	*	*	*		*	Phép trừ	R*n*←R*n*–R*m*
SUBC	R*n*	R*m*	*	*	*	*		*	Phép trừ với carry	Rn←R*n*–R*m* –c
MOV	ER*n*	ER*m*		*	*				Truyền dữ liệu (16-bit)	ER*n* ← obj
		#imm7		*	*					
ADD	ER*n*	ER*m*	*	*	*	*		*	Phép cộng (16-bit)	ER*n* ← ER*n*+obj
		#imm7	*	*	*	*		*		
CMP	ER*n*	ER*m*	*	*	*	*		*	So sánh (16-bit)	ER*n*–ER*m*

Lệnh dịch chuyển

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
SLL	R*n*	R*m*	*						Dịch chuyển logic trái theo kích thước byte	
		#width	*							
SLLC	R*n*	R*m*	*						Dịch chuyển logic trái liên tục	
		#width	*							
SRA	R*n*	R*m*	*						Dịch chuyển số học phải	
		#width	*							
SRL	R*n*	R*m*	*						Dịch chuyển logic phải	
		#width	*							
SRLC	R*n*	R*m*	*						Dịch chuyển logic phải liên tục	
		#width	*							

Lệnh Tải/Lưu

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function
L	R*n*	[EA]		*	*				Truyền dữ liệu có kích thước byte — R*n*← [EA]
		pseg_addr:[EA]		*	*				
		DSR:[EA]		*	*				
		R*d*:[EA]		*	*				
		[EA+]		*	*				Truyền dữ liệu có kích thước byte — R*n*← [EA]; EA←EA+1
		pseg_addr:[EA+]		*	*				
		DSR:[EA+]		*	*				
		R*d*:[EA+]		*	*				
		[ER*m*]		*	*				Truyền dữ liệu có kích thước byte — R*n*← [ER*m*]
		pseg_addr:[ER*m*]		*	*				
		DSR:[ER*m*]		*	*				
		R*d*:[ER*m*]		*	*				
		Disp16[ER*m*]		*	*				Truyền dữ liệu có kích thước byte — R*n*← *Disp16*[ER*m*]
		pseg_addr: *Disp16*[ER*m*]		*	*				
		DSR:*Disp16*[ER*m*]		*	*				
		R*d*:*Disp16*[ER*m*]		*	*				
		Disp6[BP]		*	*				Truyền dữ liệu có kích thước byte — R*n*← *Disp6*[BP]
		pseg_addr:*Disp6*[BP]		*	*				
		DSR: *Disp6*[BP]		*	*				
		R*d*: *Disp6*[BP]		*	*				
		Disp6[FP]		*	*				Truyền dữ liệu có kích thước byte — R*n*← *Disp6*[FP]
		pseg_addr:*Disp6*[FP]		*	*				
		DSR: *Disp6*[FP]		*	*				
		R*d*: *Disp6*[FP]		*	*				
		Dadr		*	*				Truyền dữ liệu có kích thước byte — R*n*← *Dadr*
		pseg_addr:		*	*				
		Dadr DSR:*Dadr*		*	*				
		R*d*:*Dadr*		*	*				
	ERn	[EA]		*	*				Truyền dữ liệu có kích thước word — ER*n*← [EA]
		pseg_addr:[EA]		*	*				
		DSR:[EA]		*	*				
		R*d*:[EA]		*	*				
		[EA+]		*	*				Truyền dữ liệu có kích thước word — ER*n*← [EA]; EA←EA+1
		pseg_addr:[EA+]		*	*				
		DSR:[EA+]		*	*				
		R*d*:[EA+]		*	*				
		[ER*m*]		*	*				Truyền dữ liệu có kích thước word — ER*n*← [ER*m*]
		pseg_addr:[ER*m*]		*	*				
		DSR:[ER*m*]		*	*				
		R*d*:[ER*m*]		*	*				
		Disp16[ER*m*]		*	*				Truyền dữ liệu có kích thước word — ER*n*← *Disp16*[ER*m*]
		pseg_addr: *Disp16*[ER*m*]		*	*				
		DSR:*Disp16*[ER*m*]		*	*				
		R*d*:*Disp16*[ER*m*]		*	*				
		Disp6[BP]		*	*				Truyền dữ liệu có kích thước word — ER*n*← *Disp6*[BP]
		pseg_addr:*Disp6*[BP]		*	*				
		DSR: *Disp6*[BP]		*	*				
		R*d*: *Disp6*[BP]		*	*				
		Disp6[FP]		*	*				Truyền dữ liệu có kích thước word — ER*n*← *Disp6*[FP]
		pseg_addr:*Disp6*[FP]		*	*				
		DSR: *Disp6*[FP]		*	*				
		R*d*: *Disp6*[FP]		*	*				
		Dadr		*	*				Truyền dữ liệu có kích thước word — ER*n*← *Dadr*
		pseg_addr:		*	*				
		Dadr DSR:*Dadr*		*	*				
		R*d*:*Dadr*		*	*				

(continued on next page)

Lệnh Tải/Lưu (cont.)

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
L	XR*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]		*	*				Truyền dữ liệu có kích thước double word	XR*n*← [EA]
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]		*	*				Truyền dữ liệu có kích thước double word	XR*n*← [EA] EA←EA+1
	QR*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]		*	*				Truyền dữ liệu có kích thước quad word	QR*n*← [EA]
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]		*	*				Truyền dữ liệu có kích thước quad word	QR*n*← [EA] EA←EA+1

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
ST	R*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước byte	[EA] ← R*n*
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước byte	[EA] ←R*n* EA←EA+1
		[ER*m*] *pseg_addr*:[ER*m*] DSR:[ER*m*] R*d*:[ER*m*]							Truyền dữ liệu có kích thước byte	[ER*m*] ← R*n*
		Disp16[ER*m*] *pseg_addr*: *Disp16*[ER*m*] DSR:*Disp16*[ER*m*] R*d*:*Disp16*[ER*m*]							Truyền dữ liệu có kích thước byte	*Disp16*[ER*m*] ← R*n*
		Disp6[BP] *pseg_addr*:*Disp6*[BP] DSR: *Disp6*[BP] R*d*: *Disp6*[BP]							Truyền dữ liệu có kích thước byte	*Disp6*[BP] ← R*n*
		Disp6[FP] *pseg_addr*:*Disp6*[FP] DSR: *Disp6*[FP] Rd: *Disp6*[FP]							Truyền dữ liệu có kích thước byte	*Disp6*[FP] ← R*n*
		Dadr *pseg_addr*: *Dadr* DSR:*Dadr* R*d*:*Dadr*							Truyền dữ liệu có kích thước byte	[*Dadr*] ← R*n*
	ERn	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước word	[EA] ← ER*n*
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước word	[EA] ←ER*n* EA←EA+1

Lệnh Tải/Lưu (cont.)

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
ST	ERn	[ER*m*] *pseg_addr*:[ER*m*] DSR:[ER*m*] R*d*:[ER*m*]							Truyền dữ liệu có kích thước word	[ER*m*] ← ER*n*
		Disp16[ER*m*] *pseg_addr*: *Disp16*[ER*m*] DSR:*Disp16*[ER*m*] R*d*:*Disp16*[ER*m*]							Truyền dữ liệu có kích thước word	*Disp16*[ER*m*] ←ER*n*
		Disp6[BP] *pseg_addr*:*Disp6*[BP] DSR: *Disp6*[BP] R*d*: *Disp6*[BP]							Truyền dữ liệu có kích thước word	*Disp6*[BP] ← ER*n*
		Disp6[FP] *pseg_addr*:*Disp6*[FP] DSR: *Disp6*[FP] R*d*: *Disp6*[FP]							Truyền dữ liệu có kích thước word	*Disp6*[FP] ← ER*n*
		Dadr *pseg_addr*: *Dadr* DSR:*Dadr* R*d*:*Dadr*							Truyền dữ liệu có kích thước word	[*Dadr*] ← ER*n*
	XR*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước double word	[EA] ← XR*n*
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước double word	[EA] ← XR*n* EA←EA+1
	QR*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước quad word	[EA] ← QR*n*
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước quad word	[EA] ← QR*n* EA←EA+1

Lệnh truy cập thanh ghi điều khiển

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC		Function
ADD	SP	*#signed8*							Addition	SP←SP+*signed8*
MOV	ECSR	R*m*							Truyền dữ liệu	if ELEVEL is zero LCSR ←R*m* if ELEVEL is nonzero ECSR[ELEVEL] ←R*m*
	ELR	ER*m*							Truyền dữ liệu	if ELEVEL is zero LR ← ER*m* if ELEVEL is nonzero ELR[ELEVEL] ← ER*m*
	EPSW	R*m*							Truyền dữ liệu	if ELEVEL is nonzero EPSW[ELEVEL] ←R*m*
	ER*n*	ELR							Truyền dữ liệu	if ELEVEL is zero ER*n* ←LR if ELEVEL is nonzero ER*n* ←ELR[ELEVEL]
		SP							Truyền dữ liệu	ER*n*←SP
	PSW	R*m*	*	*	*	*	*	*	Truyền dữ liệu	PSW← R*m*
		#unsigned8	*	*	*	*	*	*	Truyền dữ liệu	PSW←*unsigned8*
	R*n*	ECSR							Truyền dữ liệu	if ELEVEL is zero R*n* ← LCSR if ELEVEL is nonzero R*n* ← ECSR[ELEVEL]
		EPSW							Truyền dữ liệu	if ELEVEL is nonzero R*n* ← EPSW[ELEVEL]
		PSW							Truyền dữ liệu	R*n*←PSW
	SP	ER*m*							Truyền dữ liệu	SP←ER*m*

Lệnh PUSH/POP

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC		Function
PUSH	ER*n* R*n* QR*n* XR*n*								Lưu thanh ghi chung	SP←SP-n Stack ← General register
	register_list								Lưu thanh ghi điều khiển	SP←SP-n Stack ← Register set

POP	ER*n* R*n* QR*n* XR*n*								Khôi phục thanh ghi chung	General register ← Stack SP←SP+n
	register_list	*	*	*	*	*	*		Khôi phục thanh ghi điều khiển	Register set ← Stack*[1] SP←SP+n

*1: Thanh ghi trạng thái chương trình (PSW) chỉ thay đổi khi được đưa vào *register_list*.

Lệnh truyền dữ liệu bộ đồng xử lý

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
MOV	CR*n*	R*m*							Truyền dữ liệu có kích thước byte	CR*n* ← R*m*
	CR*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước byte	CR*n* ← [EA]
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước byte	CR*n* ← [EA+] EA ← EA+1
	CER*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước word	CER*n* ← [EA]
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước word	CER*n* ← [EA+] EA ← EA+1
	CXR*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước double word	CXR*n* ← [EA]
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước double word	CXR*n* ← [EA+] EA ← EA+1
	CQR*n*	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]							Truyền dữ liệu có kích thước quad word liên tục	CQR*n* ← [EA]
		[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]							Truyền dữ liệu có kích thước quad word liên tục	CQR*n* ← [EA+] EA ← EA+1

(continued on next page)

Lệnh truyền dữ liệu bộ đồng xử lý (continued from previous page)

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
MOV	R*n*	CR*m*							Truyền dữ liệu có kích thước byte	R*n* ←CR*m*
	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]	CR*m*							Truyền dữ liệu có kích thước byte	[EA] ←CR*m*
	[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]	CR*m*							Truyền dữ liệu có kích thước byte	[EA] ←CR*m* EA ←EA+1
	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]	CER*m*							Truyền dữ liệu có kích thước word	[EA] ←CER*m*
	[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]	CER*m*							Truyền dữ liệu có kích thước word	[EA]←CER*m* EA ←EA+1
	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]	CXR*m*							Truyền dữ liệu có kích thước double word	[EA]←CXR*m*
	[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]	CXR*m*							Truyền dữ liệu có kích thước double word	[EA] ←CXR*m* EA←EA+1
	[EA] *pseg_addr*:[EA] DSR:[EA] R*d*:[EA]	CQR*m*							Truyền dữ liệu có kích thước quad word liên tục	[EA] ←CQR*m*
	[EA+] *pseg_addr*:[EA+] DSR:[EA+] R*d*:[EA+]	CQR*m*							Truyền dữ liệu có kích thước quad word liên tục	[EA] ←CQR*m* EA←EA+1

Lệnh truyền dữ liệu thanh ghi EA

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
LEA	[ER*n*] *Disp16*[ER*m*] *Dadr*								Truyền dữ liệu tới EA	EA← ER*n* EA← *Disp16*+ER*m* EA← *Dadr*

Lệnh ALU

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
DAA	R*n*		*	*	*			*	Điều chỉnh thập phân với kích thước byte để cộng	
DAS	R*n*		*	*	*			*	Điều chỉnh thập phân với kích thước byte cho phép trừ	
NEG	R*n*		*	*	*	*		*	Phủ định	R*n* ← 0 – R*n*

Lệnh truy cập bit

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
SB	R*n.bit_offset*			*					Set bit	z← ~R*n. bit_offset* R*n. bit_offset* ←1
	Dbitadr			*					Set bit	z← ~[*Dbitadr*] [*Dbitadr*]← 1
	pseg_addr:			*						
	Dbitadr DSR:			*						
	Dbitadr			*						
	R*d*: *Dbitadr*									
RB	R*n. bit_offset*			*					Reset bit	z← ~R*n.bit_offset* R*n.bit_offset* ←0
	Dbitadr			*					Reset bit	z← ~[*Dbitadr*] [*Dbitadr*]← 0
	pseg_addr:			*						
	Dbitadr DSR:			*						
	Dbitadr			*						
	R*d*: *Dbitadr*									
TB	R*n. bit_offset*			*					Test bit	z← ~R*n.bit_offset*
	Dbitadr			*					Test bit	z← ~[*Dbitadr*]
	pseg_addr:			*						
	Dbitadr DSR:			*						
	Dbitadr			*						
	R*d*: *Dbitadr*									

Lệnh truy cập PSW

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
EI							*		Kích hoạt ngắt	MIE←1
DI							*		Vô hiệu hóa ngắt	MIE←0
SC			*						Đặt cờ nhớ	C←1
RC			*						Reset cờ nhớ	C←0
CPLC			*						Bổ sung cờ nhớ	C←–C

Lệnh nhánh quan hệ có điều kiện

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
B*cond*	*Radr*								Nhánh có điều kiện	if *cond* ? *Radr* : PC+2
BC	*cond*									

Lệnh mở rộng dấu hiệu

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC	Function	
EXTBW	ER*n*			*	*				Dấu hiệu mở rộng	ER*n*←(sign-extends)R*n*

Lệnh ngắt phần mềm

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC		Function
SWI	*#snum*						*		Lệnh ngắt phần mềm	address← (*snum*<<1), PC←Vector–table(address)
BRK									Lệnh Break	If ELEVEL greater than 1 System reset If ELEVEL less than 2 PC← (Vector–table 0004H)

Lệnh nhánh

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC		Function
B	*Cadr*								Lệnh nhánh	CSR ← *Cadr*[19:16] PC ← *Cadr*[15:0]
	ER*n*									PC ← ER*n*
BL	*Cadr*								Lệnh nhánh	LR ← Address of next instruction LCSR ← CSR CSR ← *Cadr*[19:16] PC ← *Cadr*[15:0]
	ER*n*									LR ← Address of next instruction LCSR ← CSR PC ← ER*n*

Lệnh nhân và chia

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC		Function
MUL	ER*n*	R*m*		*					Phép nhân	ER*n* ← R*n* * R*m*
DIV	ER*n*	R*m*	*	*					Phép chia	ER*n* ← ER*n*/ R*m* , R*m*←ER*n* mod R*m*

Miscellaneous

Mnemonic	First operand	Second operand	C	Z	S	OV	MIE	HC		Function
INC	[EA]			*	*	*		*	Tăng bộ nhớ	[EA] ← [EA] + 1
	pseg_addr:[EA]			*	*	*		*		
	DSR:[EA]			*	*	*		*		
	R*d*:[EA]			*	*	*		*		
DEC	[EA]			*	*	*		*	Giảm bộ nhớ	[EA] ← [EA] – 1
	pseg_addr:[EA]			*	*	*		*		
	DSR:[EA]			*	*	*		*		
	R*d*:[EA]			*	*	*		*		
RT									Trả về từ chương trình con	CSR ← LCSR PC ← LR
RTI			*	*	*	*	*	*	Trả về từ ngắt	CSR ← ECSR[ELEVEL] PC ← ELR [ELEVEL] PSW ← EPSW[ELEVEL]
NOP										

3.3 Thời gian thực thi lệnh

Phần này thảo luận về thời gian thực hiện lệnh lõi nX-U16/100. Để loại bỏ sự phụ thuộc vào tần số xung nhịp, phần này đưa ra những thời gian này theo chu kỳ xung nhịp.

Phần này cũng giả định rằng các chu kỳ đọc và ghi bộ nhớ đều dài đúng một chu kỳ xung nhịp. Tuy nhiên, trong thực tế, thời gian thực hiện lệnh truy cập bộ nhớ chậm hơn sẽ phải bao gồm các chu kỳ chờ bộ nhớ.

Mỗi lệnh cần ít nhất ba chu kỳ máy để thực thi—mỗi chu kỳ cho việc lấy lệnh, giải mã lệnh và thực thi lệnh cộng với ghi kết quả. Tuy nhiên, kiến trúc nX-U16/100 truyền các lệnh pipelines để ba giai đoạn này chạy song song, tạo ra, tối ưu, thực thi nhanh hơn so với số chu kỳ máy được đề xuất cho từng lệnh. Những thời gian thực thi này trong điều kiện tối ưu được gọi là thời gian thực thi tối thiểu.

Tuy nhiên, sự cạnh tranh về tài nguyên CPU có nghĩa là một số chuỗi lệnh nhất định không thể chạy song song. Kiến trúc nX-U16/100 giải quyết những xung đột như vậy bằng cách chèn một chu kỳ chờ ít nhất một chu kỳ máy dài vào pipeline, delay việc thực hiện lệnh sau đó.

Có ba điều kiện sau đây trong đó một chu kỳ chờ được chèn vào.

(1) Truy cập địa chỉ cửa sổ ROM sẽ tạo ra chu kỳ chờ n × m chu kỳ máy, trong đó n là số lần truy cập và m là chu kỳ chờ bộ nhớ cho một lần truy cập. Việc xử lý vùng cửa sổ ROM và số chu kỳ chờ được chèn khi vùng cửa sổ ROM được truy cập khác nhau đối với từng sản phẩm. Vui lòng tham khảo hướng dẫn sử dụng của từng sản phẩm về số chu kỳ chi tiết tại thời điểm truy cập vùng cửa sổ ROM.

(2) Khi vùng dữ liệu của phân đoạn vật lý 0 được truy cập bằng cách sử dụng địa chỉ [EA+], bus bên trong CPU sẽ cạnh tranh và nó trở thành yếu tố mà chu kỳ chờ tạo ra.

(3) Ngắt NMI và ngắt MI chịu ảnh hưởng của địa chỉ [EA+]. Các ngắt này cần 3 chu kỳ cho thời gian xử lý phần cứng, tuy nhiên, khi ngắt xảy ra ngay sau khi phân đoạn vật lý 0 được truy cập bằng cách sử dụng địa chỉ [EA+], trình tự ngắt được bắt đầu sau khi một chu kỳ máy của các chu kỳ chờ được thực hiện .

Do đó, tổng thời gian thực hiện cho một lệnh là thời gian thực hiện tối thiểu cộng với bất kỳ chu kỳ chờ nào để giải quyết xung đột bus và bất kỳ chu kỳ chờ nào của bộ nhớ.
Bảng bắt đầu ở trang tiếp theo liệt kê ba số lượng này cho tất cả các lệnh nX-U16/100. Một khoảng trống cho biết lệnh tương ứng không cạnh tranh tài nguyên CPU hoặc không truy cập bộ nhớ.

Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA+] addressing delay
ADD	ER*n*	ER*m*	1		
		#imm7	1		
ADD	R*n*	R*m*	1		
		#imm8	1		
	SP	*#signed8*	1		
ADDC	R*n*	R*m*	1		
		#imm8	1		
AND	R*n*	R*m*	1		
		#imm8	1		
B	*Cadr*		2		1
	ER*n*		2		1
B*cond*	*Radr*		A34 core :1 / 3(*1) A35 core: 1 / 2(*1)		1
BL	*Cadr*		2		1
	ER*n*		2		1
BRK			7		1
CMP	ER*n*	ER*m*	1		
	R*n*	R*m*	1		
		#imm8	1		
CMPC	R*n*	R*m*	1		
		#imm8	1		
CPLC			1		
DAA	R*n*		1		
DAS	R*n*		1		
DEC	[EA]		2		1
	pseg_addr:[EA] DSR:[EA] R*d*:[EA]		3		
DI			3		
DIV	ER*n*	R*m*	17		
EI			1		
EXTBW	ER*n*		1		
INC	[EA]		2		1
	pseg_addr:[EA] DSR:[EA] R*d*:[EA]		3		

*1: Số lượng cao hơn là khi điều kiện phân nhánh được đáp ứng; thấp hơn khi điều kiện phân nhánh không được đáp ứng.

Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA+] addressing delay
L	ER*n*	[EA]	1	1	
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	2	1	
		[EA+]	1	1	
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	2	1	
		[ER*m*]	1	1	1
		pseg_addr:[ER*m*] DSR:[ER*m*] R*d*:[ER*m*]	2	1	
		Disp16[ER*m*]	2	2	1
		pseg_addr: *Disp16*[ER*m*] DSR:*Disp16*[ER*m*] R*d*: *Disp16*[ER*m*]	4	2	
		Disp6[BP]	2	2	1
		pseg_addr: *Disp6*[BP] DSR: *Disp6*[BP] R*d*: *Disp6*[BP]	4	2	
		Disp6[FP]	2	2	1
		pseg_addr: *Disp6*[FP] DSR: *Disp6*[FP] R*d*: *Disp6*[FP]	4	2	
		Dadr	2	2	1
		pseg_addr: *Dadr* DSR: *Dadr* R*d*: *Dadr*	3	2	
	QR*n*	[EA]	4	4	
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	5	4	
		[EA+]	4	4	
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	5	4	

Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA] addressing + delay
L	R*n*	[EA]	1	1	
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	2	1	
		[EA+]	1	1	
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	2	1	
		[ER*m*]	1	1	1
		pseg_addr:[ER*m*] DSR:[ER*m*] R*d*:[ER*m*]	2	1	
		Disp16[ER*m*]	2	1	1
		pseg_addr: *Disp16*[ER*m*] DSR:*Disp16*[ER*m*] R*d*: *Disp16*[ER*m*]	3	1	
		Disp6[BP]	2	1	1
		pseg_addr: *Disp6*[BP] DSR: *Disp6*[BP] R*d*: *Disp6*[BP]	3	1	
		Disp6[FP]	2	1	1
		pseg_addr: *Disp6*[FP] DSR: *Disp6*[FP] R*d*: *Disp6*[FP]	3	1	
		Dadr	2	1	1
		pseg_addr: *Dadr* DSR: *Dadr* R*d*: *Dadr*	3	1	
	XR*n*	[EA]	2	2	
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	3	2	
		[EA+]	2	2	
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	3	2	

Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA+] addressing delay
LEA	[ER*m*]		1		
	Disp16[ER*m*]		2		
	Dadr		2		
MOV	CER*n*	[EA]	1	1	1
		pseg_addr:[EA]			
		DSR:[EA]	2	1	
		R*d*:[EA]			
		[EA+]	1	1	1
		pseg_addr:[EA+]			
		DSR:[EA+]	2	1	
		R*d*:[EA+]			
	CQR*n*	[EA]	4	4	1
		pseg_addr:[EA]			
		DSR:[EA]	5	4	
		R*d*:[EA]			
		[EA+]	4	4	1
		pseg_addr:[EA+]			
		DSR:[EA+]	5	4	
		R*d*:[EA+]			
	CR*n*	[EA]	1	1	1
		pseg_addr:[EA]			
		DSR:[EA]	2	1	
		R*d*:[EA]			
		[EA+]	1	1	1
		pseg_addr:[EA+]			
		DSR:[EA+]	2	1	
		R*d*:[EA+]			
	CR*n*	R*m*	1		
	CXR*n*	[EA]	2	2	1
		pseg_addr:[EA]			
		DSR:[EA]	3	2	
		R*d*:[EA]			
		[EA+]	2	2	1
		pseg_addr:[EA+]			
		DSR:[EA+]	3	2	
		R*d*:[EA+]			
	ECSR	R*m*	1		
	ELR	ER*m*	1		
	EPSW	R*m*	1		
	ER*n*	ELR	1		
		ER*m*	1		
		#imm7	1		
		SP	1		

Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA+] addressing delay
MOV	[EA]	CER*m*	1	1	1
	pseg_addr:[EA] DSR:[EA] R*d*:[EA]	CER*m*	2	1	
	[EA+]	CER*m*	1	1	1
	pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	CER*m*	2	1	
	[EA]	CQR*m*	4	4	1
	pseg_addr:[EA] DSR:[EA] R*d*:[EA]	CQR*m*	5	4	
	[EA+]	CQR*m*	4	4	1
	pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	CQR*m*	5	4	
	[EA]	CR*m*	1	1	1
	pseg_addr:[EA] DSR:[EA] R*d*:[EA]	CR*m*	2	1	
	[EA+]	CR*m*	1	1	1
	pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	CR*m*	2	1	
	[EA]	CXR*m*	2	2	1
	pseg_addr:[EA] DSR:[EA] R*d*:[EA]	CXR*m*	3	2	
	[EA+]	CXR*m*	2	2	1
	pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	CXR*m*	3	2	
	PSW	R*m*	1		
		#unsigned8	1		
	R*n*	CR*m*	1		
		ECSR	1		
		EPSW	1		
		PSW	1		
		R*m*	1		
		#imm8	1		
	SP	ER*m*	1		1

Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA +] addressing delay
MUL	ER*n*	R*m*	9		
NEG	R*n*		1		
NOP			1		
OR	R*n*	R*m*	1		
		#imm8	1		
POP	EA		2		1
	EA,LR		3 / 4 (*1)		1
	EA,PC		5 / 6 (*1)		1
	EA,PC,LR		6 / 8 (*1)		1
	EA,PC,PSW		6 / 7 (*1)		1
	EA,PC,PSW,LR		7 / 9 (*1)		1
	EA,PSW		3		1
	EA,PSW,LR		4 / 5 (*1)		1
	LR		1 / 2 (*1)		1
	LR,PSW		2 / 3 (*1)		1
	PC		3 / 4 (*1)		1
	PC,LR		4 / 6 (*1)		1
	PC,PSW		4 / 5 (*1)		1
	PC,PSW,LR		5 / 7 (*1)		1
	PSW		1		1
	ER*n*		1		1
	QR*n*		4		1
	R*n*		1		1
	XR*n*		2		1

*1: Số lượng thấp hơn dành cho kiểu bộ nhớ model SMALL; cao hơn, đối với model LARGE.

Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA] addressing +] dela y
PUSH	EA		1		1
	ELR		1 / 2 (*1)		1
	EA,ELR		2 / 3 (*1)		1
	EPSW		1		1
	EPSW,EA		2		1
	EPSW,ELR		2 / 3 (*1)		1
	EPSW,ELR,EA		3 / 4 (*1)		1
	LR		1 / 2 (*1)		1
	LR,EA		2 / 3 (*1)		1
	LR,ELR		2 / 4 (*1)		1
	LR,EA,ELR		3 / 5 (*1)		1
	LR,EPSW		2 / 3 (*1)		1
	LR,EPSW,EA		3 / 4 (*1)		1
	LR,EPSW,ELR		3 / 5 (*1)		1
	LR,ELR,EPSW,EA		4 / 6 (*1)		1
	ER*n*		1		1
	QR*n*		4		1
	R*n*		1		1
	XR*n*		2		1
RB	*Dbitadr*		2		1
	pseg_addr: *Dbitadr* DSR: *Dbitadr* R*d*: *Dbitadr*		3		
	R*n*. *bit_offset*		1		
RC			1		
RT			2		1
RTI			2		1
SB	*Dbitadr*		2		1
	pseg_addr: *Dbitadr* DSR: *Dbitadr* R*d*: *Dbitadr*		3		
	R*n*.*bit_offset*		1		
SC			1		

*1: Số lượng thấp hơn dành cho kiểu bộ nhớ model SMALL; cao hơn, đối với model LARGE.

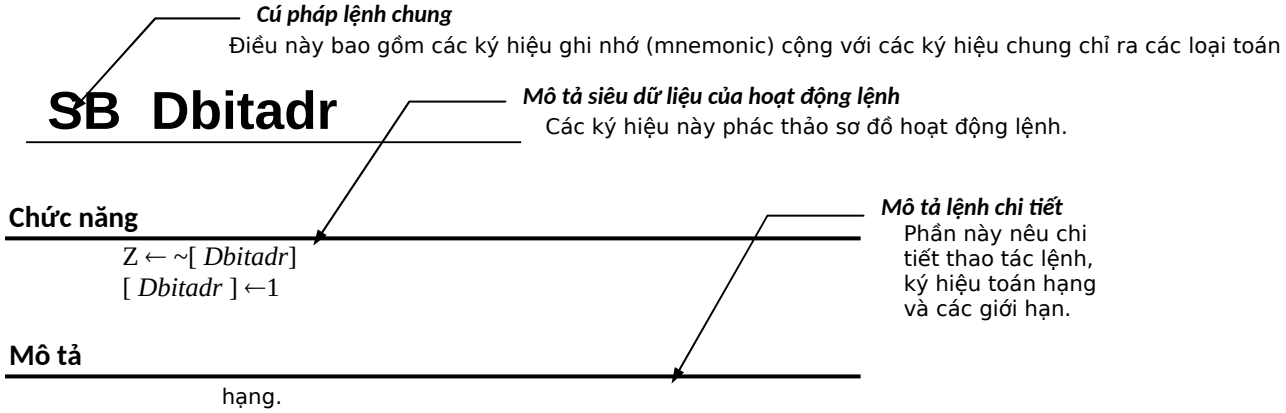
Mnemonic	First operand	Second operand	Minimum execution time (cycles)	ROM window access	[EA+] addressing delay
SLL	R*n*	R*m*	1		1
		#width	1		1
SLLC	R*n*	R*m*	1		1
		#width	1		1
SRA	R*n*	R*m*	1		1
		#width	1		1
SRL	R*n*	R*m*	1		1
		#width	1		1
SRLC	R*n*	R*m*	1		1
		#width	1		1
ST	ER*n*	[EA]	1		
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	2		
		[EA+]	1		
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	2		
		[ER*m*]	1		1
		pseg_addr:[ER*m*] DSR:[ER*m*] R*d*:[ER*m*]	2		
		Disp16[ER*m*]	2		1
		pseg_addr: *Disp16*[ER*m*] DSR:*Disp16*[ER*m*] R*d*: *Disp16*[ER*m*]	3		
		Disp6[BP]	2		1
		pseg_addr: *Disp6*[BP] DSR: *Disp6*[BP] R*d*: *Disp6*[BP]	3		
		Disp6[FP]	2		1
		pseg_addr: *Disp6*[FP] DSR: *Disp6*[FP] R*d*: *Disp6*[FP]	3		
		Dadr	2		1
		pseg_addr: *Dadr* DSR: *Dadr* R*d*: *Dadr*	3		

Mnemonic	First operand	Second operand	Minimum time execution (cycles)	ROM window access	[EA+] addressing delay
ST	QR*n*	[EA]	4		
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	5		
		[EA+]	4		
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	5		
	R*n*	[EA]	1		
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	2		
		[EA+]	1		
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	2		
		[ER*m*]	1		1
		pseg_addr:[ER*m*] DSR:[ER*m*] R*d*:[ER*m*]	2		
		Disp16[ER*m*]	2		1
		pseg_addr: *Disp16*[ER*m*] DSR:*Disp16*[ER*m*] R*d*: *Disp16*[ER*m*]	3		
		Disp6[BP]	2		1
		pseg_addr: *Disp6*[BP] DSR: *Disp6*[BP] R*d*: *Disp6*[BP]	3		
		Disp6[FP]	2		1
		pseg_addr: *Disp6*[FP] DSR: *Disp6*[FP] R*d*: *Disp6*[FP]	3		
		Dadr	2		1
		pseg_addr: *Dadr* DSR: *Dadr* R*d*: *Dadr*	3		
	XR*n*	[EA]	2		
		pseg_addr:[EA] DSR:[EA] R*d*:[EA]	3		
		[EA+]	2		
		pseg_addr:[EA+] DSR:[EA+] R*d*:[EA+]	3		

Mnemonic	First operand	Second operand	Minimum time execution (cycles)	ROM window access	[EA+] addressing delay
SUB	R*n*	R*m*	1		
SUBC	R*n*	R*m*	1		
SWI	*#snum*		3		1
TB	*Dbitadr*		2	1	1
	pseg_addr: *Dbitadr*				
	DSR: *Dbitadr*		3	1	
	R*d*: *Dbitadr*				
	R*n*. *bit_offset*		1		
XOR	R*n*	R*m*	1		
		#imm8	1		

3.4 Mô tả Lệnh

Hình sau đây mô tả cách bố trí các mô tả lệnh bắt đầu từ trang tiếp theo. Sử dụng các mẫu bit khác với các mẫu được liệt kê có thể tạo ra sự thực thi không đáng tin cậy. Các mô tả lệnh dài một hoặc hai trang với các lệnh theo thứ tự bảng chữ cái. Hình sau đây chỉ ra các phần chính của các mô tả lệnh này.



- Lệnh này kiểm tra bit được chỉ định bằng cách đọc nó từ bộ nhớ, đảo ngược nó và lưu kết quả vào cờ Z. Sau đó nó đặt bit gốc thành "1."
- Địa chỉ bit *Dbitadr* có định dạng *Dadr16.bit*, trong đó *bit* là số nguyên từ 0 đến 7 xác định vị trí bit trong byte bộ nhớ.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Bit này chuyển sang "1" nếu thao tác tạo ra kết quả bằng 0 và chuyển sang "0" nếu ngược lại.

–: Không thay đổi

Instruction Format

Mã lệnh
Bảng này liệt kê các loại địa chỉ có sẵn cho mỗi toán hạng và các mẫu bit kết quả trong mã máy cho lệnh.

Mnemonic	First operand	DSR prefix code	Instruction Format					
			First					Second word
SB	*Dbitadr*		A	0	1	*bit*	0	*Dadr*
	*: Dbitadr	<word>	A	0	1	*bit*	0	*Dadr*

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

Mã lệnh tiền tố DSR
Bảng này liệt kê các mẫu bit để sử dụng trong phần toán hạng đầu tiên được biểu thị bằng dấu hoa thị trong Bảng ngay trước đó.

ADD ERn , ERm

Add

Function

ER*n* ← ER*n* + ER*m*

Mô tả

- Lệnh này thêm nội dung của thanh ghi có kích thước second word vào thanh ghi thứ nhất và lưu trữ kết quả trong thanh ghi thứ nhất.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành "1" nếu thao tác tạo ra một carry (dư) ở bit thứ 15, và chuyển thành "0" trong trường hợp ngược lại.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra một carry ra hoặc mượn vào bit thứ 11 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
ADD	ER*n*	ER*m*	F *n* *m* 6	

ADD ERn , #imm7

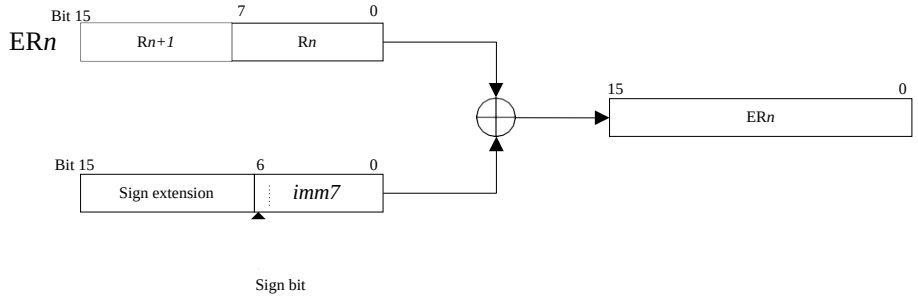
Add

Function

ER*n* ← ER*n* + (signed)*imm7*

Mô tả

- Lệnh này thêm giá trị tức thời mở rộng dấu vào nội dung của thanh ghi có kích thước word được chỉ định và lưu trữ kết quả trong thanh ghi. Hình sau đây biểu diễn hoạt động của lệnh theo sơ đồ.



Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một carry (dư) ở bit thứ 15, và chuyển thành "0" trong trường hợp ngược lại.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra một carry ra hoặc mượn vào bit thứ 11 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
ADD	ER*n*	*#imm7*	E *n* 1 *imm7*	

ADD R*n* , *obj*

Add

Function

R*n* ← R*n* + *obj*

Mô tả

- Lệnh này thêm nội dung của đối tượng có kích thước byte được chỉ định vào nội dung của thanh ghi có kích thước byte được chỉ định và lưu trữ kết quả trong thanh ghi đó.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra một carry ở bit thứ 7 và chuyển thành “0” nếu không.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra một carry ra hoặc mượn vào bit thứ 3 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format				
			First word				Second word
ADD	R*n*	R*m*	8	*n*	*m*	1	
		#imm8	1	*n*	*imm8*		

ADD SP , #*signed8*

Add

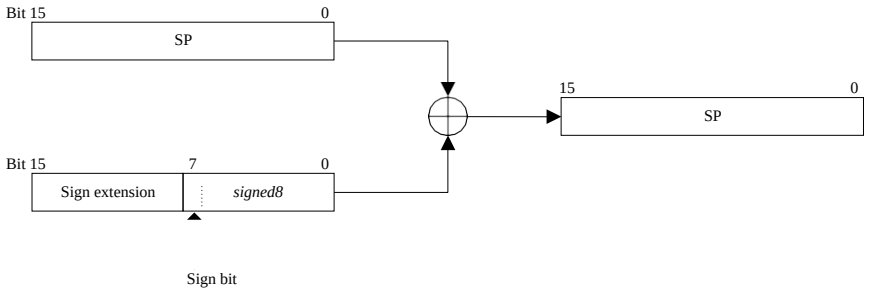
Function

SP ← SP + signed *8*

Mô tả

- Lệnh này thêm dấu mở rộng *signed8* vào nội dung của con trỏ ngăn xếp và lưu trữ kết quả trong con trỏ ngăn xếp.
- Bit 7 trong *signed8* được hiểu là bit dấu, do đó signed8 là một số nguyên nằm giữa -128 và +127.

Hình sau đây biểu diễn sơ đồ hoạt động của lệnh.



Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi.

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
ADD	SP	#*signed8*	E 1 *signed8*	

ADDC R*n* , *obj*

Add with carry

Function

R*n* ← Rn + *obj* + C

Mô tả

- Lệnh này thêm nội dung của thanh ghi có kích thước byte được chỉ định, đối tượng có kích thước byte được chỉ định và cờ nhớ C và lưu trữ kết quả trong thanh ghi.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra một carry ở bit thứ 7 và chuyển thành “0” nếu không.

Z: Cờ này vẫn là “1” chỉ khi nó là “1” trước khi thực thi và kết quả là số không. Nếu không, nó vẫn giữ nguyên hoặc

chuyển sang “0.”

S: Bit này theo dõi bit trên cùng của kết

quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một carry ra hoặc mượn vào bit thứ 3 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
ADDC	R*n*	R*m*	8 | *n* | *m* | 6	
		#imm8	6 | *n* | *imm8*	

AND R*n* , *obj*

Bitwise AND

Function

R*n* ← R*n* & *obj*

Mô tả

- Lệnh này thực hiện phép AND nội dung của thanh ghi và đối tượng có kích thước byte được chỉ định và lưu trữ kết quả trong thanh ghi.

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
AND	R*n*	R*m*	8 *n* *m* 2	
		#imm8	2 *n* *imm8*	

B Cadr

Direct branch

Function

CSR ← Cadr[19:16]
PC ← Cadr[15:0]

Mô tả

- Lệnh này nhảy đến địa chỉ được chỉ định ở bất kỳ đâu trong không gian bộ nhớ chương trình/mã.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Instruction Format				
		First word				Second word
B	*Cadr*	F	*Cadr*[19:16]	0	0	*Cadr*[15:0]

B ER*n*

Indirect branch

Function

PC ← ER*n*

Mô tả

- Lệnh này nhảy trong cùng một phân đoạn vật lý đến vị trí offset trong thanh ghi có kích thước word được chỉ định.
- Chương trình phải tải vị trí offset mục tiêu vào thanh ghi trước khi thực hiện lệnh này.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Instruction Format			
		First word			
B	ER*n*	F	0	*n*	2

Bcond Radr
BC *cond* , *Radr*

Conditional branch

Function

Nếu (*cond* = true) thì PC ← *Radr*
Lưu ý rằng khoảng cách từ địa chỉ của lệnh tiếp theo (NextPC) đến Radr phải nằm trong khoảng từ -128 đến +127.

Mô tả

- Lệnh này nhảy đến địa chỉ được chỉ định nếu nội dung thanh ghi trạng thái chương trình (PSW) thỏa mãn điều kiện được chỉ định.
- Lệnh này giả định một phép so sánh trước đó hoặc lệnh khác thiết lập cờ PSW để thử nghiệm với lệnh này.
- Có thể chỉ định điều kiện theo hai cách, một là chỉ định điều kiện như một phần của lệnh mnemonic và cách còn lại là chỉ định điều kiện như một toán hạng.

Ví dụ

```
        CMP     R0,#21H
        BEQ     LABEL     ; Điều kiện chỉ định như một phần của lệnh
        CMP     R0,#56H   ;  mnemonic.
        BC      NC,LABEL  ; Điều kiện chỉ định như toán hạng đầu tiên.
                :
                :
LBAEL:
```

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format		
			First word		
B*cond*	*Radr*		C	*condition*	(*Radr –NextPC*)>>1
BC	*cond*	*Radr*			

Condition

Instruction	Syntax			
B*cond*	BC *cond*	*Condition*	Meaning	Flag condition
BGE	BC GE	0000	Unsigned ≥	C=0
BNC	BC NC			
BLT	BC LT	0001	Unsigned <	C=1
BCY	BC CY			
BGT	BC GT	0010	Unsigned >	(C=0)&&(Z=0)
BLE	BC LE	0011	Unsigned	(Z=1)||(C=1)
BGES	BC GES	0100	≤ Signed	(OV^S)=0
BLTS	BC LTS	0101	≥ Signed <	(OV^S)=1
BGTS	BC GTS	0110	Signed >	((OV^S)| Z) = 0
BLES	BC LES	0111	Signed ≤	((OV^S)| Z) = 1
BNE	BC NE	1000	!=	Z=0
BNZ	BC NZ			
BEQ	BC EQ	1001	=	Z=1
BZ	BC ZF			
BNV	BC NV	1010	No overflow	OV=0
BOV	BC OV	1011	Overflow	OV=1
BPS	BC PS	1100	Positive	S=0
BNS	BC NS	1101	Negative	S=1
BAL	BC AL	1110	Unconditional	

BL *Cadr*

Branch and link

Function

LR ← Địa chỉ của lệnh tiếp theo
LCSR← CSR
CSR ← *Cadr*[19:16]
PC ← *Cadr*[15:0]

Mô tả

- Lệnh này lưu địa chỉ của lệnh tiếp theo trong thanh ghi liên kết (LR) và nội dung CSR hiện tại trong thanh ghi phân đoạn mã cục bộ (LCSR) rồi nhảy đến địa chỉ đã chỉ định ở bất kỳ đâu trong không gian bộ nhớ chương trình/mã.
- Lệnh này dùng để gọi một chương trình con. Để trả về từ chương trình con, hãy sử dụng lệnh RT.
- Nếu chương trình con gọi một chương trình con khác, nó phải sử dụng lệnh PUSH để lưu nội dung của các thanh ghi liên kết (LR) và đoạn mã cục bộ (LCSR) vào ngăn xếp trước lệnh gọi đầu tiên như vậy và lệnh POP để khôi phục các thanh ghi liên kết (LR) và thanh ghi phân đoạn mã cục bộ (LCSR) sau lệnh gọi cuối cùng.
- Nếu một chương trình sử dụng lệnh này trong quá trình xử lý ngắt, trình xử lý ngắt trước tiên phải sử dụng lệnh PUSH để lưu nội dung của các thanh ghi liên kết (LR) và thanh ghi phân đoạn mã cục bộ (LCSR) vào ngăn xếp trước khi gọi chương trình con và chương trình con phải quay lại với các lệnh POP tương ứng.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Instruction				
		First word				Second word
BL	*Cadr*	F	*Cadr*[19:16]	0	1	*Cadr*[15:0]

BL ER*n*

Branch and link

Function

PC ← ER*n*

LR ← Địa chỉ của lệnh tiếp theo

LCSR ← CSR

Mô tả

- Lệnh này lưu địa chỉ của lệnh tiếp theo trong thanh ghi liên kết (LR) và nội dung CSR hiện tại trong thanh ghi phân đoạn mã cục bộ (LCSR), sau đó nhảy trong cùng một đoạn vật lý đến vị trí offset trong thanh ghi có kích thước word đã chỉ định.
- Lệnh này dùng để gọi một chương trình con. Để quay lại từ chương trình con, hãy sử dụng lệnh RT.
- Nếu chương trình con gọi một chương trình con khác, nó phải sử dụng lệnh PUSH để lưu nội dung của các thanh ghi liên kết (LR) và thanh ghi phân đoạn mã cục bộ (LCSR) vào ngăn xếp trước lệnh gọi đầu tiên như vậy và lệnh POP để khôi phục các thanh ghi liên kết (LR) và thanh ghi phân đoạn mã cục bộ (LCSR) sau lệnh gọi cuối cùng.
- Nếu một chương trình sử dụng lệnh này trong trình xử lý ngắt, trình xử lý ngắt trước tiên phải sử dụng lệnh PUSH để lưu nội dung của các thanh ghi liên kết (LR) và đoạn mã cục bộ (LCSR) vào ngăn xếp trước khi gọi chương trình con và chương trình con phải trả về với các lệnh POP tương ứng.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Instruction Format			
		First word			
BL	ER*n*	F	0	*n*	3

BRK

Break instruction (software reset)

Function

- ELEVEL lớn hơn 1: System reset
- ELEVEL nhỏ hơn 2:

  ELR2 ← Địa chỉ của lệnh tiếp theo
  ECSR2 ← CSR
  EPSW2 ← PSW
  ELEVEL ← 2
  PC ← (Vector table 0004H)

Mô tả

- Lệnh này dùng để reset hệ thống ứng dụng người dùng trong phần mềm.
- ELEVEL lớn hơn 1 tạo ra lệnh thiết lập lại hệ thống CPU, which

  (1) khởi tạo tất cả các thanh ghi nội bộ CPU

  (2) tải con trỏ ngăn xếp (SP) với dữ liệu từ từ địa chỉ 0 trong không gian bộ nhớ mã/chương trình

  (3) tải bộ đếm chương trình (PC) với dữ liệu word từ địa chỉ 2 trong không gian bộ nhớ mã/chương trình
- ELEVEL nhỏ hơn 2 tạo ra tương đương với ngắt không thể vô hiệu hóa. Sau đó, CPU tải bộ đếm chương trình (PC) với dữ liệu từ từ địa chỉ bảng vectơ 4 ở đầu không gian bộ nhớ mã/chương trình.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Instruction Format			
		First word			
BRK		F	F	F	F

CMP ER*n* , ER*m*

Compare

Function

ER*n* – ER*m*

Mô tả

- Lệnh này so sánh nội dung của hai thanh ghi có kích thước word được chỉ định bằng cách trừ thanh ghi sau khỏi thanh ghi trước và đặt cờ PSW để thử nghiệm bằng nhánh có điều kiện hoặc lệnh tương tự.
- Nội dung thanh ghi không thay đổi.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một carry ra bit 15 và chuyển thành “0” nếu không.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một phép carry hoặc borrow vào bit 11 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
CMP	ER*n*	ER*m*	F ⋮ *n* ⋮ *m* ⋮ 7	

CMP R*n* , *obj*

Compare

Function

R*n* – *obj*

Mô tả

- Lệnh này so sánh nội dung của thanh ghi và đối tượng có kích thước byte được chỉ định bằng cách trừ phần sau khỏi phần trước và đặt cờ PSW để thử nghiệm bằng nhánh có điều kiện hoặc lệnh tương tự.
- Nội dung thanh ghi không thay đổi.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra một carry ở bit thứ 7 và chuyển thành “0” nếu không.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một phép thực hiện hoặc mượn vào bit 3 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
CMP	R*n*	R*m*	8 *n* *m* 7	
		#imm8	7 *n* *imm8*	

CMPC R*n* , *obj*

Compare with carry

Function

R*n* – *obj*– C

Mô tả

- Lệnh này so sánh nội dung của Rn và obj bằng cách trừ phần sau và cờ nhớ khỏi phần trước và thiết lập cờ PSW để thử nghiệm bằng nhánh có điều kiện hoặc lệnh tương tự.
- Nội dung thanh ghi không thay đổi.
- Lệnh này có thể được sử dụng sau lệnh CMP để so sánh các chuỗi nhiều byte.

Ví dụ:

```
CMP     R0, R4
CMPC    R1, R5
```

Cùng nhau, hai lệnh này so sánh các thanh ghi có kích thước word ER0 và ER4.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này chuyển thành “1” nếu phép toán tạo ra một carry ra khỏi bit 15 và chuyển thành “0” nếu không.

Z: Cờ này chỉ giữ nguyên ở “1” nếu nó đã là “1” trước khi thực thi và kết quả là zero. Nếu không, nó sẽ giữ nguyên hoặc chuyển thành “0.”

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này chuyển thành “1” nếu phép toán tạo ra một carry ra khỏi hoặc borrow vào bit 3 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
CMPC	R*n*	R*m*	8 *n* *m* 5	
		#imm8	5 *n* *imm8*	

CPLC

Complement carry flag

Function

C ← ~C

Mô tả

- Lệnh này đảo ngược nội dung của cờ nhớ.

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Đảo ngược cài đặt ban đầu

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
CPLC			F E C F	

DAA

Byte-sized decimal adjustment for addition

Function

R*n* ← (decimal adjustment) R*n*

Mô tả

- Lệnh này chuyển đổi nội dung của thanh ghi có kích thước byte được chỉ định thành giá trị thập phân mã hóa nhị phân (BCD) bằng cách thêm giá trị thích hợp, dựa trên nội dung của thanh ghi cũng như cờ C và HC, từ Bảng sau. “X” chỉ ra rằng CPU không quan tâm đến nội dung của phần đó.

C	R*n*[7:4]	HC	R*n*[3:0]	Adjustment	C flag after adjustment
0	0–9	0	0–9	00	0
0	0–8	0	A–F	06	0
0	0–9	1	x	06	0
0	A–F	0	0–9	60	1
0	9–F	0	A–F	66	1
0	A–F	1	x	66	1
1	x	0	0–9	60	1
1	x	0	A–F	66	1
1	x	1	x	66	1

- Một lệnh cộng nhị phân (ADD Rn, obj) phải đi trước lệnh này và bất kỳ lệnh xen kẽ nào cũng không được thay đổi nội dung của thanh ghi hoặc thanh ghi trạng thái chương trình (PSW).

Flags

C	Z	S	OV	MIE	HC
*	*	*	–	–	*

C: Cờ này chuyển thành “1” nếu quá trình thực thi tạo ra một carry vào vị trí 100s. Nếu không, nó sẽ giữ nguyên.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

HC: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra phép carry hoặc borrow vào bit 3 và chuyển thành “0” nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
DAA	R*n*		8 *n* 1 F	

DAS

Byte-sized decimal adjustment for subtraction

Function

R*n* ← (decimal adjustment) R*n*

Mô tả

- Lệnh này chuyển đổi nội dung của thanh ghi có kích thước byte được chỉ định thành giá trị thập phân được mã hóa nhị phân (BCD) bằng cách trừ giá trị thích hợp, dựa trên nội dung của thanh ghi cũng như cờ C và HC, từ Bảng sau..

"X" cho biết CPU không quan tâm đến nội dung của phần đó.

C	R*n*[7:4]	HC	R*n*[3:0]	Adjustment
0	0–9	0	0–9	00
0	0–9	0	A–F	06
0	0–9	1	x	06
0	A–F	0	0–9	60
0	A–F	1	x	66
0	A–F	0	A–F	66
1	x	0	0–9	60
1	x	1	x	66
1	x	0	A–F	66

- Một lệnh trừ nhị phân (SUB Rn, obj) phải đi trước lệnh này và bất kỳ lệnh xen kẽ nào cũng không được thay đổi nội dung của thanh ghi hoặc thanh ghi trạng thái chương trình (PSW).

Flags

C	Z	S	OV	MIE	HC
*	*	*	–	–	*

C: Cờ này chuyển thành "1" nếu thực hiện tạo ra một borrow từ vị trí 100s. Nếu không, nó vẫn không thay đổi.

Z: Bit này sẽ chuyển thành "1" nếu phép toán tạo ra kết quả bằng không và chuyển thành "0" nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

HC: Bit này sẽ chuyển thành "1" nếu thao tác tạo ra một carry hoặc borrow vào bit 3 và chuyển thành "0" nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
DAS	R*n*		8 *n* 3 F	

DEC [EA]

Memory decrement
(using EA indirect addressing)

Function

[EA] ← [EA] –1

Mô tả

- Lệnh này trừ một từ byte tại địa chỉ trong thanh ghi EA.

Flags

C	Z	S	OV	MIE	HC
–	*	*	*	–	*

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một carry hoặc borrow vào bit 3 và chuyển thành “0” nếu không.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format	
				First word	Second word
DEC	[EA]			F E 3 F	
	*:[EA]		<word>	F E 3 F	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

DI

Disable interrupts

Function

MIE ← 0

Mô tả

- Lệnh này đặt bit cho phép ngắt chính (MIE) thành “0” để vô hiệu hóa các ngắt có thể vô hiệu hóa.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	*	–

MIE: Được chuyển thành “0.”

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
DI			E B F 7	

DIV ERn , Rm

Division

Function

ERn ← ER*n* / R*m*
Rm ← ER*n* mod R*m*

Mô tả

- Lệnh này chia nội dung của thanh ghi có kích thước word được chỉ định cho nội dung của thanh ghi có kích thước byte được chỉ định, lưu trữ số chia 16 bit trong thanh ghi trước và lưu trữ số dư 8 bit trong thanh ghi sau.
- Bộ chia số không đặt cờ nhớ thành "1" và để lại các giá trị không xác định trong cả hai thanh ghi.

Flags

C	Z	S	OV	MIE	HC
*	*	–	–	–	–

C: Cờ này sẽ chuyển thành "1" nếu số chia bằng 0. Nếu không, nó sẽ chuyển thành "0."
Z: Cờ này sẽ chuyển thành "1" nếu số bị chia bằng 0. Nếu không, nó sẽ chuyển thành "0."
–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
DIV	ER*n*	R*m*	F *n* *m* 9	

EI

Enable interrupts

Function

MIE ← 1

Mô tả

- Lệnh này đặt bit cho phép ngắt chính (MIE) thành “1” để cho phép ngắt có thể vô hiệu hóa.
- Lưu ý rằng bit MIE không chuyển sang “1” trong ba chu kỳ kể từ khi bắt đầu lệnh này, do đó chương trình ứng dụng của người dùng phải hỗ trợ ngắt có thể vô hiệu hóa trong hai chu kỳ sau lệnh này.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	*	–

MIE: Chuyển thành “1.”

–: Không thay đổi

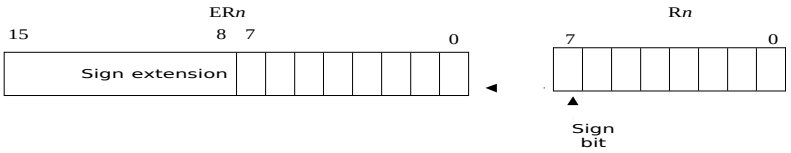
Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
EI			E D 0 8	

EXTBW ER*n*

Extend sign

Function



Mô tả

- Lệnh này mở rộng nội dung của thanh ghi R*n* thành định dạng 16 bit có dấu và lưu trữ trong thanh ghi ER*n*.
- Nội dung của R*n*+1 được điền bằng bit 7 của thanh ghi R*n* , như kết quả.

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Bit này sẽ chuyển thành “1” nếu giá trị thanh ghi R*n* bằng không và chuyển thành “0” nếu không.

S: Bit này theo dõi bit 7 của thanh ghi R*n*.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
EXTBW	ER*n*		8 *n*+1 *n* F	

INC [EA]

Memory increment
(using EA indirect addressing)

Function

[EA] ← [EA] +1

Mô tả

- Lệnh này thêm một vào byte tại địa chỉ trong thanh ghi EA.

Flags

C	Z	S	OV	MIE	HC
–	*	*	*	–	*

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.
S: Bit này theo dõi bit trên cùng của kết quả.
OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.
HC: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một carry hoặc borrow vào bit 3 và chuyển thành “0” nếu không.
–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction First word	Instruction Second word	Format
INC	[EA]			F E 2 F		
	*:[EA]		<word>	F E 2 F		

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

L ER*n*, *obj*

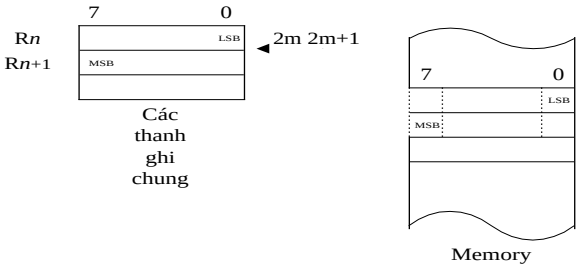
Word-sized data transfer

Function

ER*n* ← *obj*

Mô tả

- Lệnh này tải thanh ghi 16 bit được chỉ định với dữ liệu tại địa chỉ word được chỉ định.



Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

(Xem trang tiếp theo)

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format	
				First word	Second word
L	ER*n*	[EA]		9 n 3 2	
		*: [EA]	<word>	9 n 3 2	
		[EA+]		9 *n* 5 2	
		*:[EA+]	<word>	9 *n* 5 2	
		[ER*m*]		9 *n* *m* 2	
		*:[ER*m*]	<word>	9 *n* *m* 2	
		Disp16[ER*m*]		A *n* *m* 8	*Disp16*
		*:*Disp16*[ER*m*]	<word>	A *n* *m* 8	*Disp16*
		Disp6[BP]		B *n* 0 0 *Disp6*	
		*:*Disp6*[BP]	<word>	B *n* 0 0 *Disp6*	
		Disp6[FP]		B *n* 0 1 *Disp6*	
		*:*Disp6*[FP]	<word>	B *n* 0 1 *Disp6*	
		Dadr		9 *n* - 1 2	*Dadr*
		*: *Dadr*	<word>	9 *n* 1 2	*Dadr*

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

L

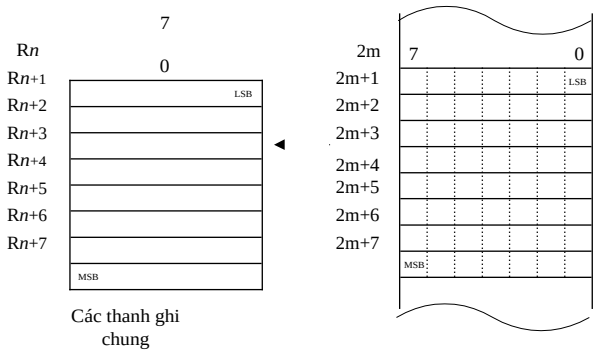
Quad word-sized data transfer

Function

QR*n* ← *obj*

Mô tả

- Lệnh này tải thanh ghi 64 bit được chỉ định với dữ liệu tại địa chỉ word được chỉ định.



Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Cờ này sẽ chuyển thành “1” nếu nội dung mới của thanh ghi là số không. Nếu không, nó sẽ chuyển thành “0.”

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Second operan	DSR prefix code	Instruction Format				
				First wor				Secon
L	QR*n*	[EA]		9	*n*	3	6	
		*: [EA]	<word>	9	*n*	3	6	
		[EA+]		9	*n*	5	6	
		*:[EA+]	<word>	9	*n*	5	6	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

L R*n*, *obj*

Truyền dữ liệu có kích thước byte

Function

R*n* ←*obj*

Mô tả

- Lệnh này tải thanh ghi 8 bit được chỉ định với dữ liệu tại địa chỉ byte được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Cờ này sẽ chuyển thành “1” nếu nội dung mới của thanh ghi là số không. Nếu không, nó sẽ chuyển thành “0.”

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

(Xem trang tiếp theo)

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format				
				First word				Second word
L	R*n*	[EA]		9	n	3	0	
		*: [EA]	<word>	9	n	3	0	
		[EA+]		9	*n*	5	0	
		*:[EA+]	<word>	9	*n*	5	0	
		[ERm]		9	*n*	*m*	0	
		*:[ERm]	<word>	9	*n*	*m*	0	
		Disp16[ER*m*]		9	*n*	*m*	8	*Disp16*
		*:*Disp16*[ER*m*]	<word>	9	*n*	*m*	8	*Disp16*
		Disp6[BP]		D	*n*	0 0	*Disp6*	
		*:*Disp6*[BP]	<word>	D	*n*	0 0	*Disp6*	
		Disp6[FP]		D	*n*	0 1	*Disp6*	
		*:*Disp6*[FP]	<word>	D	*n*	0 1	*Disp6*	
		Dadr		9	*n*	1	0	*Dadr*
		*: *Dadr*	<word>	9	*n*	1	0	*Dadr*

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

L

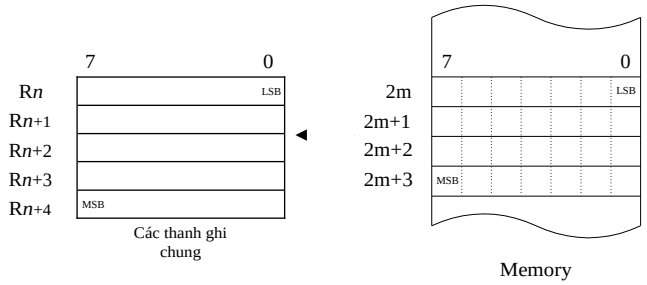
Double word-sized data transfer

Function

XR*n* ← *obj*

Mô tả

- Lệnh này tải thanh ghi 32 bit được chỉ định với dữ liệu tại địa chỉ word được chỉ định.



Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Cờ này sẽ chuyển thành “1” nếu nội dung mới của thanh ghi là số không. Nếu không, nó sẽ chuyển thành “0.”

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Second operan	DSR prefix code	Instruction Format				
				First wor				Secon
L	XR*n*	[EA]		9	*n*	3	4	
		*: [EA]	<word>	9	*n*	3	4	
		[EA+]		9	*n*	5	4	
		*:[EA+]	<word>	9	*n*	5	4	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

LEA *obj*

Load EA

Function

EA ← *obj*

Mô tả

- Lệnh này tải thanh ghi EA với giá trị word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format				
			First word				Second word
LEA	[ER*m*]		F	0	*m*	A	
	Dadr		F	0	0	C	*Dadr*
	Disp16[ER*m*]		F	0	*m*	B	*Disp16*

MOV CER*n* , *obj*

Coprocessor data transfer

Function

CER*n* ←*obj*

Mô tả

- Lệnh này tải thanh ghi có kích thước word của bộ đồng xử lý được chỉ định từ địa chỉ word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format				
				First word				Second word
MOV	CER*n*	[EA]		F	*n*	2	D	
		*: [EA]	<word>	F	*n*	2	D	
		[EA+]		F	*n*	3	D	
		*:[EA+]	<word>	F	*n*	3	D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV CQR*n* , *obj*

Coprocessor data transfer

Function

CQR*n* ← *obj*

Mô tả

- Lệnh này tải thanh ghi có kích thước quad word của bộ đồng xử lý được chỉ định từ địa chỉ word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format	
				First word	Second word
MOV	CQR*n*	[EA]		F *n* 6 D	
		*: [EA]	<word>	F *n* 6 D	
		[EA+]		F *n* 7 D	
		*:[EA+]	<word>	F *n* 7 D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV CR*n* , *obj*

Coprocessor data transfer

Function

CR*n* ←*obj*

Mô tả

- Lệnh này tải thanh ghi có kích thước byte của bộ đồng xử lý được chỉ định từ địa chỉ byte được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format				
				First word				Second word
MOV	CR*n*	[EA]		F	*n*	0	D	
		*: [EA]	<word>	F	*n*	0	D	
		[EA+]		F	*n*	1	D	
		*:[EA+]	<word>	F	*n*	1	D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV CR*n* , R*m*

Coprocessor data transfer

Function

CR*n* ←R*m*

Mô tả

- Lệnh này tải thanh ghi có kích thước byte của bộ đồng xử lý được chỉ định từ thanh ghi nội bộ có kích thước byte được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	CR*n*	R*m*	A | *n* | *m* | E	

MOV CXR*n* , *obj*

Coprocessor data transfer

Function

CXR*n* ← *obj*

Mô tả

- Lệnh này tải thanh ghi có kích thước double word của bộ đồng xử lý được chỉ định từ thanh ghi nội bộ có kích thước double word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format				
				First word				Second word
MOV	CXR*n*	[EA]		F	*n*	4	D	
		*: [EA]	<word>	F	*n*	4	D	
		[EA+]		F	*n*	5	D	
		*:[EA+]	<word>	F	*n*	5	D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV ECSR , R*m*

Data transfer

Function

- Nếu ELEVEL là không
  LCSR ←R*m*
- Nếu ELEVEL khác không
  ECSR[ELEVEL] ←R*m*

Mô tả

- Lệnh này tải nội dung của thanh ghi được chỉ định vào thanh ghi đoạn mã cục bộ (LCSR) nếu ELEVEL bằng không , nếu ELEVEL không bằng không thì sẽ vào thanh ghi ECSR (ECSR1 đến ECSR3) để thiết lập mức ngoại lệ hiện tại (ELEVEL).

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	ECSR	R*m*	A 0 *m* F	

MOV ELR , ER*m*

Data transfer

Function

- Nếu ELEVEL là không
  LR ← ER*m*
- Nếu ELEVEL khác không
  ELR[ELEVEL] ← ER*m*

Mô tả

- Lệnh này tải nội dung của thanh ghi có kích thước word được chỉ định vào thanh ghi liên kết (LR) nếu ELEVEL bằng không và vào thanh ghi liên kết ngoại lệ (ELR1 đến ELR3) cho cài đặt mức ngoại lệ hiện tại (ELEVEL) nếu ELEVEL không bằng không.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	ELR	ER*m*	A *m* 0 D	

MOV EPSW , R*m*

Data transfer

Function

- Nếu ELEVEL khác không
  EPSW[ELEVEL] ←R*m*

Mô tả

- Lệnh này tải nội dung của thanh ghi được chỉ định vào thanh ghi sao lưu trạng thái chương trình (EPSW1 đến EPSW3) cho cài đặt mức ngoại lệ hiện tại (ELEVEL) nếu ELEVEL khác không.
- Nếu ELEVEL bằng không, lệnh này không làm gì cả. Bộ đếm chương trình (PC) chỉ đơn giản là chuyển sang lệnh tiếp theo.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	EPSW	R*m*	A | 0 | *m* | C	

MOV ER*n* , ELR

Data transfer

Function

- Nếu ELEVEL là không
  ER*n* ←LR
- Nếu ELEVEL khác không
  ER*n* ←ELR[ELEVEL]

Mô tả

- Lệnh này tải thanh ghi có kích thước word được chỉ định từ thanh ghi liên kết (LR) nếu ELEVEL bằng không và từ thanh ghi liên kết ngoại lệ (ELR1 đến ELR3) cho cài đặt mức ngoại lệ hiện tại (ELEVEL) nếu ELEVEL không bằng không.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	ER*n*	ELR	A *n* 0 5	

MOV ER*n* , ER*m*

Data transfer

Function

ER*n* ←ER*m*

Mô tả

- Lệnh này tải thanh ghi có kích thước word đầu tiên từ thanh ghi thứ hai.

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Cờ này sẽ chuyển thành “1” nếu nội dung mới của thanh ghi là số không. Nếu không, nó sẽ chuyển thành “0”.

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

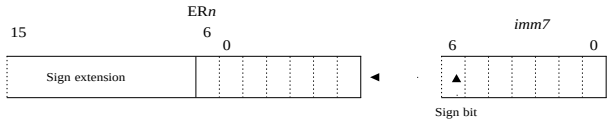
Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	ER*n*	ER*m*	F *n* *m* 5	

MOV ERn , #imm7

Data transfer

Function

ER*n* ← (sign–extends)*imm7*



Mô tả

- Lệnh này tải dấu mở rộng *imm7* vào thanh ghi có kích thước word được chỉ định. Chính xác hơn, nó tải giá trị tức thời vào R*n*, nửa dưới của thanh ghi, và sao chép bit 6 từ giá trị tức thời vào bit 7 R*n* và tất cả các bit của R*n*+1.

Example:

```
MOV     R0,#07Fh
MOV     R1,#0h
MOV     ER0,#-64      ; Thực hiện sao chép bit trên cùng (“1”), đặt R0 thành 0C0H
                        và R1 thành 0FFH

MOV     R0,#03Fh
MOV     R1,#0FFh
MOV     ER0,#3Fh      ; Thực hiện sao chép bit trên cùng (“0”), đặt R0 thành 03FH
                        và R1 thành 0H
```

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Cờ này sẽ chuyển thành “1” nếu nội dung mới của thanh ghi là số không. Nếu không, nó sẽ chuyển thành “0”.

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	ER*n*	*#imm7*	E *n* 0 *imm7*	

MOV ERn , SP

Data transfer

Function

ER*n* ←SP

Mô tả

- Lệnh này lưu nội dung của con trỏ ngăn xếp (SP) trong thanh ghi có kích thước word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	ER*n*	SP	A *n* 1 A	

MOV *obj* , CER*m*

Coprocessor data transfer

Function

(WORD) *obj* ←CER*m*

Mô tả

- Lệnh này lưu nội dung của thanh ghi có kích thước word của bộ đồng xử lý được chỉ định tại địa chỉ word được chỉ định trong thanh ghi EA.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format	
				First word	Second word
MOV	[EA]	CER*m*		F *m* A D	
	* : [EA]	CER*m*	<word>	F *m* A D	
	[EA+]	CER*m*		F *m* B D	
	* : [EA+]	CER*m*	<word>	F *m* B D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV *obj* , CQR*m*

Coprocessor data transfer

Function

(QWORD)*obj* ←CQR*m*

Mô tả

- Lệnh này lưu nội dung của thanh ghi có kích thước quad word của bộ đồng xử lý được chỉ định tại địa chỉ word được chỉ định trong thanh ghi EA.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Second operan	DSR prefix code	Instruction Format				
				First wor				Secon
MOV	[EA]	CQR*m*		F	*m*	E	D	
	*: [EA]	CQR*m*	<word>	F	*m*	E	D	
	[EA+]	CQR*m*		F	*m*	F	D	
	*: [EA+]	CQR*m*	<word>	F	*m*	F	D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV *obj* , CR*m*

Coprocessor data transfer

Function

(BYTE) *obj* ←CR*m*

Mô tả

- Lệnh này lưu nội dung của thanh ghi có kích thước byte của bộ đồng xử lý được chỉ định tại địa chỉ byte được chỉ định trong thanh ghi EA.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Second operan	DSR prefix code	Instruction Format				
				First wor				Secon
MOV	[EA]	CR*m*		F	*m*	8	D	
	*: [EA]	CR*m*	<word>	F	*m*	8	D	
	[EA+]	CR*m*		F	*m*	9	D	
	*: [EA+]	CR*m*	<word>	F	*m*	9	D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV *obj* , CXR*m*

Coprocessor data transfer

Function

(DOUBLE WORD)*obj* ← CXR*m*

Mô tả

- Lệnh này lưu nội dung của thanh ghi có kích thước double word của bộ đồng xử lý được chỉ định tại địa chỉ word được chỉ định trong thanh ghi EA.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Second operan	DSR Prefix code	Instruction Format First wor				Secon
MOV	[EA]	CXR*m*		F	*m*	C	D	
	*: [EA]	CXR*m*	<word>	F	*m*	C	D	
	[EA+]	CXR*m*		F	*m*	D	D	
	*: [EA+]	CXR*m*	<word>	F	*m*	D	D	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

MOV PSW , *obj*

Data transfer

Function

PSW ← *obj*

Mô tả

- Lệnh này tải thanh ghi trạng thái chương trình (PSW) từ đối tượng có kích thước byte được chỉ định.
- Khi mức ngoại lệ hiện tại (ELEVEL) thay đổi, cần phải sắp xếp lệnh NOP ngay sau đó. Nếu không, lệnh sau sẽ hoạt động trước khi thay đổi ELEVEL và kết quả là chương trình có thể bị trục trặc.

  Example:

  ```
  MOV PSW, #05h
  NOP
  RTI
  ```

- Khi giá trị của bit cho phép ngắt chính (MIE) được đặt lại thành 0, lệnh DI được sử dụng và lệnh này không được sử dụng. Nếu không, bit MIE không chuyển sang “0” trong ba chu kỳ kể từ khi bắt đầu lệnh này, do đó, ngắt có thể vô hiệu hóa mà lập trình viên không mong muốn và có khả năng chương trình ứng dụng bị trục trặc.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	*	*

*: Nội dung phản ánh bit nguồn tương ứng.

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format				
			First word				Second word
MOV	PSW	*#unsigned8*	E	9	*unsigned8*		
		R*m*	A	0	*m*	B	

MOV R*n* , CR*m*

Coprocessor data transfer

Function

R*n* ← CR*m*

Mô tả

- Lệnh này tải thanh ghi có kích thước byte được chỉ định từ thanh ghi có kích thước byte của bộ đồng xử lý được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	R*n*	CR*m*	A | *n* | *m* | 6	

MOV R*n* , ECSR

Data transfer

Function

- Nếu ELEVEL là không

  R*n* ← LCSR

- If ELEVEL khác không

  R*n* ← ECSR[ELEVEL]

Mô tả

- Lệnh này tải thanh ghi có kích thước byte được chỉ định từ thanh ghi đoạn mã cục bộ (LCSR) nếu ELEVEL bằng không và từ thanh ghi ECSR (ECSR1 đến ECSR3) cho cài đặt mức ngoại lệ hiện tại (ELEVEL) nếu ELEVEL không bằng không.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	R*n*	ECSR	A *n* 0 7	

MOV R*n* , EPSW

Data transfer

Function

- Nếu ELEVEL khác không
  R*n* ← EPSW[ELEVEL]

Mô tả

- Lệnh này tải thanh ghi có kích thước byte được chỉ định từ thanh ghi sao lưu trạng thái chương trình (EPSW1 đến EPSW3) cho cài đặt mức ngoại lệ hiện tại (ELEVEL) nếu ELEVEL khác không.
- Nếu ELEVEL bằng không, lệnh này tải 0xFF.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	R*n*	EPSW	A *n* 0 4	

MOV R*n* , PSW

Data transfer

Function

R*n* ← PSW

Mô tả

- Lệnh này tải thanh ghi có kích thước byte được chỉ định từ thanh ghi trạng thái chương trình (PSW).

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	R*n*	PSW	A | *n* | 0 | 3	

MOV R*n* , *obj*

Data transfer

Function

Rn ←*obj*

Mô tả

- Lệnh này tải thanh ghi có kích thước byte được chỉ định từ đối tượng có kích thước byte được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Cờ này sẽ chuyển thành “1” nếu nội dung mới của thanh ghi là số không. Nếu không, nó sẽ chuyển thành “0”.

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	R*n*	R*m*	8 | *n* | *m* | 0	
		#imm8	0 | *n* | *imm8*	

MOV SP , ER*m*

Data transfer

Function

SP ← ER*m*

Mô tả

- Lệnh này tải con trỏ ngăn xếp (SP) từ thanh ghi có kích thước word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MOV	SP	ER*m*	A 1 *m* A	

MUL ERn,Rm

Multiplication

Function

ER*n* ← R*n* * R*m* (*n* phải đều)

Mô tả

- Lệnh này nhân nội dung của hai thanh ghi kích thước byte được chỉ định và lưu trữ tích 16 bit trong thanh ghi kích thước word tương ứng với thanh ghi đầu tiên.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
MUL	ER*n*	R*m*	F *n* *m* 4	

NEG R*n*

Negate

Function

R*n* ←0–R*n*

Mô tả

- Lệnh này tính toán phần bù hai của nội dung của thanh ghi kích thước byte được chỉ định và lưu trữ kết quả trong thanh ghi đó.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra một carry ở bit thứ 7 và chuyển thành “0” nếu không.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra carry hoặc borrow vào bit 3 và chuyển thành “0” nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
NEG	R*n*		8 *n* 5 F	

NOP

No operation

Function

Không có hoạt động

Mô tả

- Lệnh này đưa bộ đếm chương trình (PC) tới lệnh tiếp theo.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
NOP			F E 8 F	

OR Rn , *obj*

Bitwise OR

Function

R*n* ←R*n* | *obj*

Mô tả

- Lệnh này OR nội dung của thanh ghi và đối tượng có kích thước byte được chỉ định và lưu trữ kết quả trong thanh ghi.

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Cờ này sẽ chuyển thành “1” nếu nội dung mới của thanh ghi là số không. Nếu không, nó sẽ chuyển thành “0”.

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
OR	R*n*	R*m*	8 *n* *m* 3	
		#imm8	3 *n* *imm8*	

POP register list

Restore control registers

Function

Các thanh ghi điều khiển ← (SP)

SP ←SP + n

Mô tả

- Lệnh này tải các thanh ghi điều khiển được chỉ định từ ngăn xếp hệ thống được trỏ tới bởi con trỏ ngăn xếp (SP) và sau đó tăng SP theo số byte tương ứng. Để biết thêm chi tiết, hãy xem Phần 1.7 “Sửa đổi ngăn xếp.”
- Các thanh ghi điều khiển sau đây có thể xuất hiện trong danh sách này.
  (1) Thanh ghi EA
  (2) thanh ghi liên kết (LR) để lưu bộ đếm chương trình (PC) khi gọi một chương trình con
  (3) thanh ghi trạng thái chương trình (PSW)
  (4) bộ đếm chương trình (PC)
- Danh sách này không cần phải chứa tất cả, nhưng nó phải chứa ít nhất một.
- Danh sách này có thể xuất hiện theo bất kỳ thứ tự nào, nhưng phần cứng luôn sử dụng thứ tự được đưa ra bên dưới:

$$EA \rightarrow LR \rightarrow PSW \rightarrow PC$$

- Quá trình thông thường để trả về từ một trình xử lý ngắt hoặc chương trình con là với lệnh RT hoặc RTI, nhưng đôi khi cần phải lưu nội dung của các thanh ghi sao lưu vào ngăn xếp với lệnh PUSH khi các trình xử lý ngắt hoặc chương trình con được lồng vào nhau và khôi phục chúng bằng lệnh POP sau đó. Để biết thêm chi tiết, hãy xem Mục 1.4 “Mức độ ngoại lệ và Thanh ghi dự phòng.”

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	*	*

*: Nội dung chỉ thay đổi nếu PSW có trong danh sách.

Instruction Format

Mnemonic	operand	Instruction			
		First word			
POP	EA	F	1	8	E
	PC	F	2	8	E
	EA, PC	F	3	8	E
	PSW	F	4	8	E
	EA, PSW	F	5	8	E
	PC, PSW	F	6	8	E
	EA, PC, PSW	F	7	8	E
	LR	F	8	8	E
	EA, LR	F	9	8	E
	PC, LR	F	A	8	E
	EA, PC, LR	F	B	8	E
	LR, PSW	F	C	8	E
	EA, PSW, LR	F	D	8	E
	PC, PSW, LR	F	E	8	E
	EA, PC, PSW, LR	F	F	8	E

POP *obj*

Restore general registers

Function

Các thanh ghi chung ← (SP)

SP ←SP + n

Mô tả

- Lệnh này tải các thanh ghi chung được chỉ định từ ngăn xếp hệ thống được trỏ tới bởi con trỏ ngăn xếp (SP) khi nó tăng SP theo số byte tương ứng.
- Vì các hoạt động ngăn xếp luôn có kích thước word, nên lệnh này với số byte toán hạng lẻ (R*n*) sẽ tải thanh ghi được chỉ định và đọc một byte giả mà không sửa đổi bất kỳ thanh ghi nào khác.

Để biết thêm chi tiết, hãy xem Phần 1.7 “Sửa đổi ngăn xếp.”

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Instruction Format First
POP	R*n*	F *n* 0 0 0 0 E
	ER*n* QR*n* XR*n*	F *n* 0 0 0 1 E F *n* 0 0 1 0 E

PUSH register list

Save control registers

Function

SP ←SP – n

(SP) ← Các thanh ghi điều khiển

Mô tả

- Lệnh này lưu các thanh ghi điều khiển được chỉ định vào ngăn xếp hệ thống được trỏ tới bởi con trỏ ngăn xếp (SP) khi nó giảm SP theo số byte tương ứng. Để biết thêm chi tiết, hãy xem Phần 1.7 “Sửa đổi ngăn xếp.”
- Các thanh ghi điều khiển sau đây có thể xuất hiện trong danh sách này.
  (1) thanh ghi liên kết ngoại lệ (ELR)
  (2) thanh ghi trạng thái chương trình ngoại lệ (EPSW)
  (3) thanh ghi liên kết (LR) để lưu bộ đếm chương trình (PC) khi gọi một chương trình con
  (4) thanh ghi EA
- Danh sách này có thể xuất hiện theo bất kỳ thứ tự nào, nhưng phần cứng luôn sử dụng thứ tự được đưa ra bên dưới:

$$ELR \rightarrow EPSW \rightarrow LR \rightarrow EA$$

- Lệnh này giả định rằng các Lệnh PUSH trước đó đã lưu các thanh ghi điều khiển được chỉ định trên ngăn xếp theo thứ tự thích hợp.
- Quá trình thông thường để trả về từ một trình xử lý ngắt hoặc chương trình con là với lệnh RT hoặc RTI, nhưng đôi khi cần phải lưu nội dung của các thanh ghi sao lưu vào ngăn xếp với lệnh PUSH khi các trình xử lý ngắt hoặc chương trình con được lồng vào nhau và khôi phục chúng bằng lệnh POP sau đó. Để biết thêm chi tiết, hãy xem Mục 1.4 “Mức độ ngoại lệ và Thanh ghi dự phòng.”

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Instruction Format First			
PUSH	EA	F	1	C	E
	ELR	F	2	C	E
	EA, ELR	F	3	C	E
	EPSW	F	4	C	E
	EPSW, EA	F	5	C	E
	EPSW, ELR	F	6	C	E
	EPSW, ELR, EA	F	7	C	E
	LR	F	8	C	E
	LR, EA	F	9	C	E
	LR, ELR	F	A	C	E
	LR, EA, ELR	F	B	C	E
	LR, EPSW	F	C	C	E
	LR, EPSW, EA	F	D	C	E
	LR, EPSW, ELR	F	E	C	E
	LR, EPSW, ELR, EA	F	F	C	E

PUSH *obj*

Save general registers

Function

SP ←SP – n

(SP) ← Các thanh ghi chung

Mô tả

- Lệnh này tải các thanh ghi chung được chỉ định từ ngăn xếp hệ thống được trỏ tới bởi con trỏ ngăn xếp (SP) khi nó giảm SP theo số byte tương ứng.
- Vì các hoạt động ngăn xếp luôn có kích thước word, nên lệnh này với toán hạng số byte lẻ (R*n*) lưu trữ thanh ghi được chỉ định bằng một byte giả.
  Để biết thêm chi tiết, hãy xem Phần 1.7 “Sửa đổi ngăn xếp.”

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Instruction Format
		First word
PUSH	R*n*	F *n* 0100 E
	ER*n*	F *n* 0101 E
	XR*n*	F *n* 0110 E
	QR*n*	F *n* 0111 E

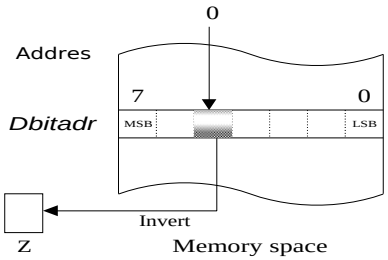
RB Dbitadr

Reset bit

Function

Z ← ~[*Dbitadr*]

[*Dbitadr*] ← 0



Mô tả

- Lệnh này kiểm tra bit được chỉ định bằng cách đọc bit đó từ bộ nhớ, đảo ngược bit đó và lưu trữ kết quả trong cờ Z. Sau đó, lệnh này đặt lại bit gốc thành "0."
- Địa chỉ bit *Dbitadr* có định dạng *Dadr.bit*, trong đó bit là số nguyên từ 0 đến 7 chỉ định vị trí bit trong byte bộ nhớ.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Đảo ngược của bit gốc

–: Không thay đổi

Instruction Format

Mnemonic	First operand	DSR prefix code	Instruction Format	
			First word	Second word
RB	*Dbitadr*		A 0 1 *bit* 2	*Dadr*
	*: Dbitadr	<word>	A 0 *bit* 2	*Dadr*

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

RB R*n* . *bit_offset*

Reset bit

Function

Z ← ~Rn[*bit_offset*]

Rn[*bit_offset*] ← 0

Mô tả

- Lệnh này đọc bit được chỉ định từ thanh ghi có kích thước byte được chỉ định, đảo ngược bit đó và lưu trữ trong cờ Z. Sau đó, lệnh này đặt lại bit gốc thành "0."
- *bit_offset* là một số nguyên từ 0 đến 7 chỉ định vị trí bit trong thanh ghi.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Đảo ngược của bit gốc

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
RB	R*n.bit_offset*		A *n* 0 *bit* 2	

RC

Reset carry flag

Function

C ← 0

Mô tả

- Lệnh này reset cờ nhớ về “0.”

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Bit này sẽ chuyển thành “0.”

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
RC			E B 7 F	

RT

Return from subroutine

Function

CSR ← LCSR
PC ← LR

Mô tả

- Lệnh này dùng để trả về từ một chương trình con được gọi bằng lệnh BL. Lệnh này khôi phục địa chỉ của lệnh theo lệnh BL bằng cách tải thanh ghi phân đoạn mã từ thanh ghi phân đoạn mã cục bộ (LCSR) và bộ đếm chương trình (PC) từ thanh ghi liên kết (LR).

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
RT			F E 1 F	

RTI

Return from interrupt

Function

CSR ← ECSR[ELEVEL]

PC ← ELR [ELEVEL]

PSW ← EPSW[ELEVEL]

Mô tả

- Lệnh này dùng để trả về từ trình xử lý ngắt. Lệnh này khôi phục thanh ghi trạng thái chương trình (PSW) và bộ đếm chương trình (PC) từ thanh ghi sao lưu trạng thái chương trình (EPSW1 đến EPSW3) và thanh ghi liên kết ngoại lệ (ELR1 đến ELR3), tương ứng, cho thiết lập mức ngoại lệ hiện tại (ELEVEL) —1 cho các ngắt có thể vô hiệu hóa và 2 cho các ngắt không thể vô hiệu hóa.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	*	*

*: Nội dung phản ánh bit EPSW tương ứng.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
RTI			F E 0 F	

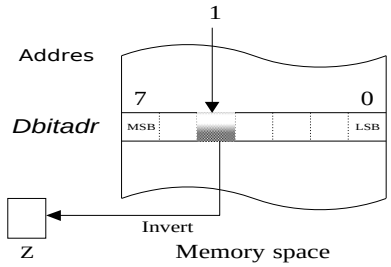
SB Dbitadr

Set bit

Function

Z ← ~[*Dbitadr*]

[*Dbitadr*] ←1



Mô tả

- Lệnh này kiểm tra bit được chỉ định bằng cách đọc bit đó từ bộ nhớ, đảo ngược bit đó và lưu trữ kết quả trong cờ Z. Sau đó, lệnh này đặt bit gốc thành "1."
- Địa chỉ bit *Dbitadr* có định dạng *Dadr16.bit*, trong đó bit là số nguyên từ 0 đến 7 chỉ định vị trí bit trong byte bộ nhớ.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Bit này sẽ chuyển thành "1" nếu phép toán tạo ra kết quả bằng không và chuyển thành "0" nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	DSR prefix code	Instruction Format					
			First word					Second word
SB	*Dbitadr*		A	0	1	*bit*	0	*Dadr*
	*: *Dbitadr*	<word>	A	0	1	*bit*	0	*Dadr*

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

SB Rn . *bit_offset*

Set bit

Function

Z ← ~R*n*[*bit_offset*]

Rn[*bit_offset*] ← 1

Mô tả

- Lệnh này đọc bit được chỉ định từ thanh ghi có kích thước byte được chỉ định, đảo ngược bit đó và lưu trữ bit đó trong cờ Z. Sau đó, lệnh này đặt bit gốc thành "1."
- *bit_offset* là số nguyên từ 0 đến 7 chỉ định vị trí bit trong thanh ghi.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Bit này sẽ chuyển thành "1" nếu phép toán tạo ra kết quả bằng không và chuyển thành "0" nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
SB	R*n*.*bit_offset*		A *n* 0 bit 0	

SC

Set carry flag

Function

C ← 1

Mô tả

- Lệnh này đặt cờ nhớ thành “1.”

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Bit này sẽ chuyển thành “1.”

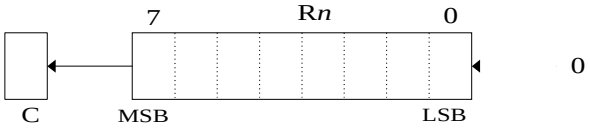
Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
SC			E D 8 0	

SLL R*n* , *obj*

Shift left logical

Function



Mô tả

- Lệnh này dịch chuyển các bit trong thanh ghi có kích thước byte được chỉ định sang trái số vị trí được chỉ định bởi toán hạng thứ hai và dịch chuyển các số không từ bên phải. Cờ nhớ giữ lại bit cuối cùng được dịch chuyển ra ngoài.
- Phạm vi có ý nghĩa cho kích thước dịch chuyển là từ 0 đến 7. Nếu toán hạng thứ hai là thanh ghi có kích thước byte, phần cứng sẽ bỏ qua các bit từ 7 đến 3 trong thanh ghi đó và chỉ sử dụng ba bit thấp nhất, do đó hạn chế kích thước dịch chuyển trong phạm vi từ 0 đến 7. Kích thước dịch chuyển bằng 0 tạo ra tương đương với lệnh NOP. Đặt trước lệnh này bằng một chuỗi lệnh SLLC cho phép thực hiện thao tác dịch chuyển trên các chuỗi bit dài hơn trong nhiều thanh ghi. (Xem ví dụ SLLC.)

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Bit này giữ lại bit cuối cùng đã dịch chuyển ra ngoài.

–: Không thay đổi

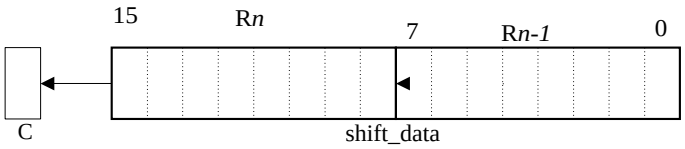
Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
SLL	R*n*	R*m*	8 *n* *m* A	
		#width	9 *n* 0 *width* A	

SLLC R*n* , *obj*

Shift left logical continued

Function



Mô tả

- Lệnh này dịch chuyển 16 bit trong thanh ghi có kích thước byte được chỉ định và thanh ghi bên dưới nó (hoặc R15 nếu R0 được chỉ định) để lại số vị trí được chỉ định bởi toán hạng thứ hai (tối đa là 7 vị trí) và lưu trữ tám bit trên cùng trong thanh ghi được chỉ định. Cờ nhớ giữ lại bit cuối cùng được dịch chuyển ra ngoài.
- Phạm vi có ý nghĩa cho kích thước dịch chuyển là từ 0 đến 7. Nếu toán hạng thứ hai là thanh ghi có kích thước byte, phần cứng sẽ bỏ qua các bit từ 7 đến 3 trong thanh ghi đó và chỉ sử dụng ba bit thấp nhất, do đó hạn chế kích thước dịch chuyển trong phạm vi từ 0 đến 7. Kích thước dịch chuyển bằng 0 tạo ra tương đương với lệnh NOP.
- Một chuỗi các lệnh này theo sau là một lệnh SLL cho phép thực hiện thao tác dịch chuyển trên các chuỗi bit dài hơn trong nhiều thanh ghi. (Xem ví dụ.)

Example: Dịch chuyển sang trái cho dữ liệu double word

```
 SLLC  R3, R5
SLLC  R2, R5
SLLC  R1, R5
SLL    R0, R5        Việc này hoàn tất việc chuyển đổi nội dung XR0
```

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Bit này giữ lại bit cuối cùng đã dịch chuyển ra ngoài.

–: Không thay đổi

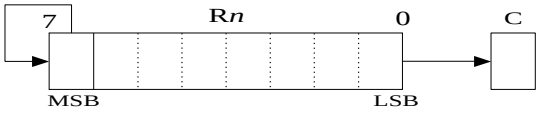
Instruction Format

Mnemonic	First operand	Second operand	Instruction Format					
			First word					Second word
SLLC	R*n*	R*m*	8	*n*		*m*	B	
		#width	9	*n*	0	*width*	B	

SRA R*n* , *obj*

Shift right arithmetic

Function



Mô tả

- Lệnh này dịch chuyển các bit trong thanh ghi có kích thước byte được chỉ định sang phải số vị trí được chỉ định bởi toán hạng thứ hai và dịch chuyển các bản sao của bit dấu gốc (bit 7) từ bên trái. Cờ nhớ giữ lại bit cuối cùng được dịch chuyển ra ngoài.
- Phạm vi có ý nghĩa cho kích thước dịch chuyển là từ 0 đến 7. Nếu toán hạng thứ hai là thanh ghi có kích thước byte, phần cứng sẽ bỏ qua các bit từ 7 đến 3 trong thanh ghi đó và chỉ sử dụng ba bit thấp nhất, do đó hạn chế kích thước dịch chuyển trong phạm vi từ 0 đến 7. Kích thước dịch chuyển bằng 0 tạo ra tương đương với lệnh NOP.

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Bit này giữ lại bit cuối cùng đã dịch chuyển ra ngoài.

–: Không thay đổi

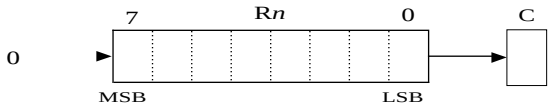
Instruction Format

	Mnemonic	First Second	Instruction Format				
			First word				Second
SRA	R*n*	R*m*	8	*n*	*m*	E	
		#width	9	*n*	0 *width*	E	

SRL R*n* , *obj*

Shift right logical

Function



Mô tả

- Lệnh này dịch chuyển các bit trong thanh ghi có kích thước byte được chỉ định sang phải số vị trí được chỉ định bởi toán hạng thứ hai và dịch chuyển các số không từ bên trái. Cờ nhớ giữ lại bit cuối cùng được dịch chuyển ra ngoài.
- Phạm vi có ý nghĩa cho kích thước dịch chuyển là từ 0 đến 7. Nếu toán hạng thứ hai là thanh ghi có kích thước byte, phần cứng sẽ bỏ qua các bit từ 7 đến 3 trong thanh ghi đó và chỉ sử dụng ba bit thấp nhất, do đó hạn chế kích thước dịch chuyển trong phạm vi từ 0 đến 7. Kích thước dịch chuyển bằng 0 tạo ra tương đương với lệnh NOP. Đặt trước lệnh này bằng một chuỗi lệnh SRLC cho phép thực hiện thao tác dịch chuyển trên các chuỗi bit dài hơn trong nhiều thanh ghi. (Xem ví dụ về SRLC.)

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Bit này giữ lại bit cuối cùng đã dịch chuyển ra ngoài.

–: Không thay đổi

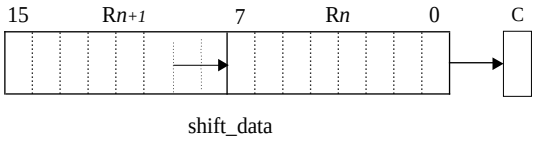
Instruction Format

	Mnemonic	First Second	Instruction Format					
			First wor					Secon
SRL	R*n*	R*m*	8	*n*		*m*	C	
		#width	9	*n*	0	*width*	C	

SRLC R*n* , *obj*

Shift right logical continued

Function



shift_data

Mô tả

- Lệnh này dịch chuyển 16 bit trong thanh ghi có kích thước byte được chỉ định và thanh ghi ở trên nó (hoặc R0 nếu R15 được chỉ định) sang phải số vị trí được chỉ định bởi toán hạng thứ hai (tối đa là 7 vị trí) và lưu trữ tám bit thấp hơn trong thanh ghi được chỉ định. Cờ nhớ giữ lại bit cuối cùng được dịch chuyển ra ngoài.
- Phạm vi có ý nghĩa cho kích thước dịch chuyển là từ 0 đến 7. Nếu toán hạng thứ hai là một thanh ghi có kích thước byte, phần cứng sẽ bỏ qua các bit từ 7 đến 3 trong thanh ghi đó và chỉ sử dụng ba bit thấp nhất, do đó hạn chế kích thước dịch chuyển trong phạm vi từ 0 đến 7. Kích thước dịch chuyển bằng 0 tạo ra lệnh NOP tương đương.

Example: Dịch chuyển sang phải cho dữ liệu double word

```
 SRLC  R0, R5
SRLC  R1, R5
SRLC  R2, R5
SRL    R3, R5        Việc này hoàn tất việc dịch chuyển nội dung XR0
```

Flags

C	Z	S	OV	MIE	HC
*	–	–	–	–	–

C: Bit này giữ lại bit cuối cùng đã dịch chuyển ra ngoài.

–: Không thay đổi

Instruction Format

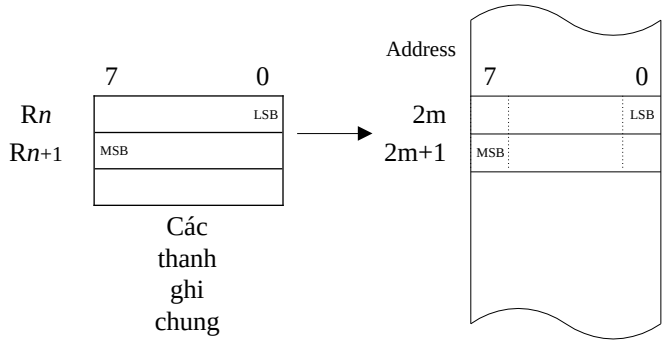
	Mnemonic	First Second	Instruction Format				
			First wor				Secon
SRLC	R*n*	R*m*	8	*n*	*m*	D	
		#width	9	*n*	0 *width*	D	

ST ERn , obj

Word-sized data transfer

Function

obj ← ER*n*



Mô tả

- Lệnh này lưu trữ nội dung của thanh ghi 16 bit được chỉ định tại địa chỉ word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

(Xem trang tiếp theo)

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format	
				First word	Second word
ST	ER*n*	[EA]		9 *n* 3 3	
		*: [EA]	<word>	9 *n* 3 3	
		[EA+]		9 *n* 5 3	
		*:[EA+]	<word>	9 *n* 5 3	
		[ER*m*]		9 *n* *m* 3	
		*:[ER*m*]	<word>	9 *n* *m* 3	
		Disp16[ER*m*]		A *n* *m* 9	*Disp16*
		*:*Disp16*[ER*m*]	<word>	A *n* *m* 9	*Disp16*
		Disp6[BP]		B *n* 1 0 *Disp6*	
		*:*Disp6*[BP]	<word>	B *n* 1 0 *Disp6*	
		Disp6[FP]		B *n* 1 1 *Disp6*	
		*:*Disp6*[FP]	<word>	B *n* 1 1 *Disp6*	
		Dadr		9 *n* 1 3	*Dadr*
		*: *Dadr*	<word>	9 *n* 1 3	*Dadr*

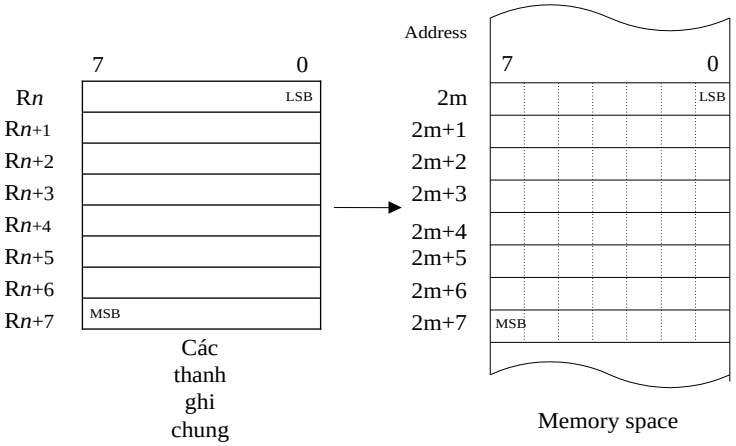
*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

ST QR*n* ,

Quad word-sized data transfer

Function

obj ← QR*n*



Mô tả

- Lệnh này lưu trữ nội dung của thanh ghi 64 bit được chỉ định tại địa chỉ word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Second operan	DSR prefix code	Instruction Format				
				First wor				Secon
ST	QR*n*	[EA]		9	*n*	3	7	
		*: [EA]	<word>	9	*n*	3	7	
		[EA+]		9	*n*	5	7	
		*:[EA+]	<word>	9	*n*	5	7	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

ST R*n* , *obj*

Byte-sized data transfer

Function

obj ← R*n*

Mô tả

- Lệnh này lưu trữ nội dung của thanh ghi 8 bit được chỉ định tại địa chỉ được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

(Xem trang tiếp theo)

Mnemonic	First operand	Second operand	DSR prefix code	Instruction Format				
				First word				Second word
ST	R*n*	[EA]		9	n	3	1	
		*: [EA]	<word>	9	n	3	1	
		[EA+]		9	*n*	5	1	
		*:[EA+]	<word>	9	*n*	5	1	
		[ER*m*]		9	*n*	*m*	1	
		*:[ER*m*]	<word>	9	*n*	*m*	1	
		Disp16[ER*m*]		9	*n*	*m*	9	*Disp16*
		*:*Disp16*[ER*m*]	<word>	9	*n*	*m*	9	*Disp16*
		Disp6[BP]		D	*n*	10	*Disp6*	
		*:*Disp6*[BP]	<word>	D	*n*	10	*Disp6*	
		Disp6[FP]		D	*n*	11	*Disp6*	
		*:*Disp6*[FP]	<word>	D	*n*	11	*Disp6*	
		Dadr		9	*n*	1	1	*Dadr*
		**: Dadr*	<word>	9	*n*	1	1	*Dadr*

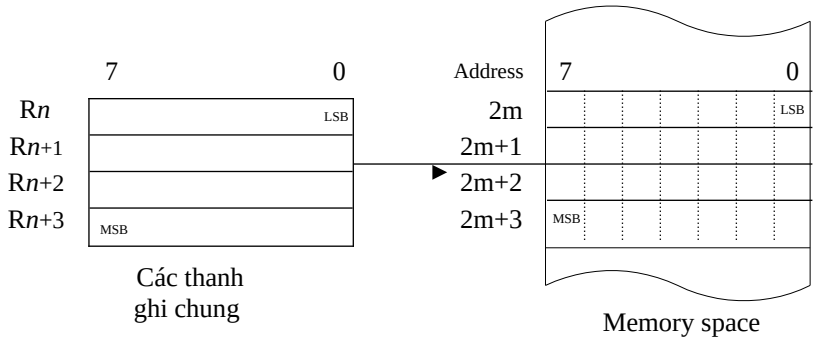
*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

ST XR*n* ,

Double word-sized data transfer

Function

obj ← XR*n*



Mô tả

- Lệnh này lưu trữ nội dung của thanh ghi 32 bit được chỉ định tại địa chỉ word được chỉ định.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	–	–

–: Không thay đổi

Instruction Format

Mnemonic	First operan	Second operan	DSR prefix	Instruction Format				
				First wor				Secon
ST	XR*n*	[EA]		9	*n*	3	5	
		*: [EA]	<word>	9	*n*	3	5	
		[EA+]		9	*n*	5	5	
		*:[EA+]	<word>	9	*n*	5	5	

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

SUB R*n* , R*m*

Subtract

Function

R*n* ← R*n* – R*m*

Mô tả

- Lệnh này trừ nội dung của thanh ghi có kích thước byte thứ hai khỏi nội dung của thanh ghi thứ nhất và lưu trữ kết quả trong thanh ghi thứ nhất.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một phép mượn vào bit 7 và chuyển thành “0” nếu không.

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra carry hoặc borrow vào bit 3 và chuyển thành “0” nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
SUB	R*n*	R*m*	8 | *n* | *m* | 8	

SUBC R*n* , R*m*

Subtract with carry

Function

R*n* ← R*n* – R*m* – C

Mô tả

- Lệnh này trừ nội dung của thanh ghi có kích thước byte thứ hai và cờ nhớ khỏi nội dung của thanh ghi đầu tiên và lưu trữ kết quả trong thanh ghi đầu tiên.

Flags

C	Z	S	OV	MIE	HC
*	*	*	*	–	*

C: Bit này sẽ chuyển thành “1” nếu thao tác tạo ra một phép mượn vào bit 7 và thành “0” nếu ngược lại.

Z: Cờ này vẫn giữ nguyên “1” chỉ khi nó là “1” trước khi thực thi và kết quả là không. Nếu ngược lại, nó vẫn giữ nguyên

hoặc chuyển thành “0.”

S: Bit này theo dõi bit trên cùng của kết

quả.

OV: Bit này sẽ chuyển thành “1” nếu hoạt động tạo ra tràn và chuyển thành “0” nếu không.

HC: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra carry hoặc borrow vào bit 3 và chuyển thành “0” nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
SUBC	R*n*	R*m*	8 | *n* | *m* | 9	

SWI *#snum*

Software interrupt

Function

EPSW1 ←PSW
ELEVEL ← 1
ELR1 ← PC+2
ECSR1 ←CSR
MIE ←0
PC ←TABLE[*snum*<<1]

Mô tả

- Lệnh này tải mục nhập bảng vectơ được chỉ định vào bộ đếm chương trình (PC). Toán hạng là một số nguyên từ 0 đến 63. Trong chu kỳ ngắt, lệnh này cũng lưu địa chỉ của lệnh tiếp theo trong thanh ghi ELR1.

Flags

C	Z	S	OV	MIE	HC
–	–	–	–	*	–

MIE: Nó sẽ thành “0.”
–: Không thay đổi

Instruction Format

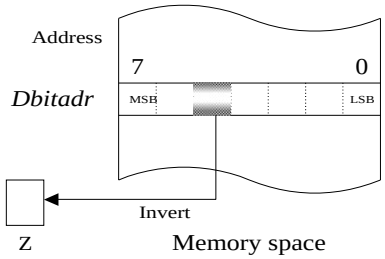
Mnemonic	First operand	Instruction					
		First word					Second word
SWI	*#snum*	E	5	0	0	*snum*	

TB Dbitadr

Test bit

Function

Z ← ~[*Dbitadr*]



Mô tả

- Lệnh này kiểm tra bit được chỉ định bằng cách đọc nó từ bộ nhớ, đảo ngược nó và lưu trữ kết quả trong cờ Z.
- Địa chỉ bit *Dbitadr* có định dạng *Dadr16.bit*, trong đó *bit* là số nguyên từ 0 đến 7 chỉ định vị trí bit trong byte bộ nhớ.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	DSR prefix code	Instruction Format					
			First word					Second word
TB	*Dbitadr*		A	0	1	*bit*	1	*Dadr*
	*: *Dbitadr*	<word>	A	0	1	*bit*	1	*Dadr*

*	<word>			
pseg_addr	E	3	*pseg_addr*	
DSR	F	E	9	F
R*d*	9	0	*d*	F

TB Rn . bit_offset

Test bit

Function

Z ← ~R*n*[bit_offset]

Mô tả

- Lệnh này kiểm tra bit được chỉ định bằng cách đọc nó từ bộ nhớ, đảo ngược nó và lưu trữ kết quả trong cờ Z.
- *bit_offset* là một số nguyên từ 0 đến 7 chỉ định vị trí bit trong thanh ghi.

Flags

C	Z	S	OV	MIE	HC
–	*	–	–	–	–

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
TB	R*n*.bit		A *n* 0 bit 1	

XOR R*n* , *obj*

Bitwise exclusive OR

Function

R*n* ←R*n*^ *obj*

Mô tả

- Lệnh này thực hiện phép XOR nội dung của thanh ghi và đối tượng có kích thước byte được chỉ định và lưu trữ kết quả trong thanh ghi.

Flags

C	Z	S	OV	MIE	HC
–	*	*	–	–	–

Z: Bit này sẽ chuyển thành “1” nếu phép toán tạo ra kết quả bằng không và chuyển thành “0” nếu ngược lại.

S: Bit này theo dõi bit trên cùng của kết quả.

–: Không thay đổi

Instruction Format

Mnemonic	First operand	Second operand	Instruction Format	
			First word	Second word
XOR	R*n*	R*m*	8 *n* *m* 4	
		#imm8	4 *n* *imm8*	

4. Phụ lục

TPhụ lục này liệt kê các lệnh cốt lõi nX-U16/100 theo nhóm chức năng, cung cấp cú pháp toán hạng và mã lệnh cho từng lệnh.

Phần mô tả về các lệnh tiền tố DSR bị lược bỏ trong chương này. Do đó, vui lòng tham khảo Chương 3 để biết thông tin chi tiết về từng lệnh.

Lệnh thuật toán

Mnemoni	Firs operand	Secon operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time (cycles)
ADD	R*n*	R*m*	*	*	*	*		*	1000_*nnnn*_*mmmm*_0001		1
		#imm8	*	*	*	*		*	0001_*nnnn*_iiii_iiii		1
ADD	ER*n*	ER*m*	*	*	*	*		*	1111_*nnn*0_*mmm*0_0110		1
		#imm7	*	*	*	*		*	1110_*nnn*0_1iii_iiii		1
ADDC	R*n*	R*m*	*	*	*	*		*	1000_*nnnn*_*mmmm*_0110		1
		#imm8	*	*	*	*		*	0110_*nnnn*_iiii_iiii		1
AND	R*n*	R*m*		*	*				1000_*nnnn*_*mmmm*_0010		1
		#imm8		*	*				0010_*nnnn*_iiii_iiii		1
CMP	R*n*	R*m*	*	*	*	*		*	1000_*nnnn*_*mmmm*_0111		1
		#imm8	*	*	*	*		*	0111_*nnnn*_iiii_iiii		1
CMPC	R*n*	R*m*	*	*	*	*		*	1000_*nnnn*_*mmmm*_0101		1
		#imm8	*	*	*	*		*	0101_*nnnn*_iiii_iiii		1
MOV	ER*n*	ER*m*		*	*				1111_*nnn*0_*mmm*0_0101		1
		#imm7		*	*				1110_*nnn*0_0iii_iiii		1
MOV	R*n*	R*m*		*	*				1000_*nnnn*_*mmmm*_0000		1
		#imm8		*	*				0000_*nnnn*_iiii_iiii		1
OR	R*n*	R*m*		*	*				1000_*nnnn*_*mmmm*_0011		1
		#imm8		*	*				0011_*nnnn*_iiii_iiii		1
XOR	R*n*	R*m*		*	*				1000_*nnnn*_*mmmm*_0100		1
		#imm8		*	*				0100_*nnnn*_iiii_iiii		1
CMP	ER*n*	ER*m*	*	*	*	*		*	1111_*nnn*0_*mmm*0_0111		1
SUB	R*n*	R*m*	*	*	*	*		*	1000_*nnnn*_*mmmm*_1000		1
SUBC	R*n*	R*m*	*	*	*	*		*	1000_*nnnn*_*mmmm*_1001		1

Lệnh dịch chuyển

Mnemoni	First operand	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
SLL	R*n*	R*m*	*						1000_*nnnn*_*mmmm*_1010		1
		#width	*						1001_*nnnn*_0www_1010		1
SLLC	R*n*	R*m*	*						1000_*nnnn*_*mmmm*_1011		1
		#width	*						1001_*nnnn*_0www_1011		1
SRA	R*n*	R*m*	*						1000_*nnnn*_*mmmm*_1110		1
		#width	*						1001_*nnnn*_0www_1110		1
SRL	R*n*	R*m*	*						1000_*nnnn*_*mmmm*_1100		1
		#width	*						1001_*nnnn*_0www_1100		1
SRLC	R*n*	R*m*	*						1000_*nnnn*_*mmmm*_1101		1
		#width	*						1001_*nnnn*_0www_1101		1

Lệnh Tải/Lưu

Mnemonic	First operand	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
L	ER*n*	[EA]		*	*				1001_*nnn*0_0011_0010		1
		[EA+]		*	*				1001_*nnn*0_0101_0010		1
		[ER*m*]		*	*				1001_*nnn*0_*mmm*0_0010		1
		Disp16[ER*m*]		*	*				1010_*nnn*0_*mmm*0_1000	DDDD_DDDD_DDDD_DDDD	2
		Disp6[BP]		*	*				1011_*nnn*0_00DD_DDDD		2
		Disp6[FP]		*	*				1011_*nnn*0_01DD_DDDD		2
		Dadr		*	*				1001_*nnn*0_0001_0010	DDDD_DDDD_DDDD_DDDD	2
	R*n*	[EA]		*	*				1001_*nnnn*_0011_0000		1
		[EA+]		*	*				1001_*nnnn*_0101_0000		1
		[ER*m*]		*	*				1001_*nnnn*_*mmm*0_0000		1
		Disp16[ER*m*]		*	*				1001_*nnnn*_*mmm*0_1000	DDDD_DDDD_DDDD_DDDD	2
		Disp6[BP]		*	*				1101_*nnnn*_00DD_DDDD		2
		Disp6[FP]		*	*				1101_*nnnn*_01DD_DDDD		2
		Dadr		*	*				1001_*nnnn*_0001_0000	DDDD_DDDD_DDDD_DDDD	2
	XR*n*	[EA]		*	*				1001_*nn*00_0011_0100		2
		[EA+]		*	*				1001_*nn*00_0101_0100		2
	QR*n*	[EA]		*	*				1001_*n*000_0011_0110		4
		[EA+]		*	*				1001_*n*000_0101_0110		4

Mnemonic	First operand	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
ST	ER*n*	[EA]							1001_*nnn*0_0011_0011		1
		[EA+]							1001_*nnn*0_0101_0011		1
		[ER*m*]							1001_*nnn*0_*mmm*0_0011		1
		Disp16[ER*m*]							1010_*nnn*0_*mmm*0_1001	DDDD_DDDD_DDDD_DDDD	2
		Disp6[BP]							1011_*nnn*0_10DD_DDDD		2
		Disp6[FP]							1011_*nnn*0_11DD_DDDD		2
		Dadr							1001_*nnn*0_0001_0011	DDDD_DDDD_DDDD_DDDD	2
	R*n*	[EA]							1001_*nnnn*_0011_0001		1
		[EA+]							1001_*nnnn*_0101_0001		1
		[ER*m*]							1001_*nnnn*_*mmm*0_0001		1
		Disp16[ER*m*]							1001_*nnnn*_*mmm*0_1001	DDDD_DDDD_DDDD_DDDD	2
		Disp6[BP]							1101_*nnnn*_10DD_DDDD		2
		Disp6[FP]							1101_*nnnn*_11DD_DDDD		2
		Dadr							1001_*nnnn*_0001_0001	DDDD_DDDD_DDDD_DDDD	2
	XR*n*	[EA]							1001_*nn*00_0011_0101		2
		[EA+]							1001_*nn*00_0101_0101		2
	QR*n*	[EA]							1001_*n*000_0011_0111		4
		[EA+]							1001_*n*000_0101_0111		4

Lệnh truy cập thanh ghi điều khiển

Mnemoni	First operand	Second	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
ADD	SP	*#signed8*							1110_0001_iiii_iiii		1
MOV	ECSR	R*m*							1010_0000_*mmmm*_1111		1
	ELR	ER*m*							1010_*mmm*0_0000_1101		1
	EPSW	R*m*							1010_0000_*mmmm*_1100		1
	ER*n*	ELR							1010_*nnn*0_0000_0101		1
		SP							1010_*nnn*0_0001_1010		1
	PSW	R*m*	*	*	*	*	*	*	1010_0000_*mmmm*_1011		1
		#unsigned8	*	*	*	*	*	*	1110_1001_iiii_iiii		1
	R*n*	ECSR							1010_*nnnn*_0000_0111		1
		EPSW							1010_*nnnn*_0000_0100		1
		PSW							1010_*nnnn*_0000_0011		1
	SP	ER*m*							1010_0001_*mmm*0_1010		1

Lệnh PUSH/POP

Mnemoni	First operan	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
PUSH	ER*n*								1111_*nnn*0_0101_1110		1
	QR*n*								1111_*n*000_0111_1110		4
	R*n*								1111_*nnnn*_0100_1110		1
	XR*n*								1111_*nn*00_0110_1110		2
	register_list								1111_lepa_1100_1110		1-6
POP	ER*n*								1111_*nnn*0_0001_1110		1
	QR*n*								1111_*n*000_0011_1110		4
	R*n*								1111_*nnnn*_0000_1110		1
	XR*n*								1111_*nn*00_0010_1110		2
	register_list		*	*	*	*	*	*	1111_lepa_1000_1110		1-9

Lệnh truyền dữ liệu bộ đồng xử lý

Mnemonic	First operand	Second operand	Flag changes C Z S OV MIE HC	Instruction code First word	Second	Minimum execution time
MOV	CR*n*	R*m*		1010_*nnnn*_*mmmm*_1110		1
	CER*n*	[EA]		1111_*nnn*0_0010_1101		1
		[EA+]		1111_*nnn*0_0011_1101		1
	CR*n*	[EA]		1111_*nnnn*_0000_1101		1
		[EA+]		1111_*nnnn*_0001_1101		1
	CXR*n*	[EA]		1111_*nn*00_0100_1101		2
		[EA+]		1111_*nn*00_0101_1101		2
	CQR*n*	[EA]		1111_*n*000_0110_1101		4
		[EA+]		1111_*n*000_0111_1101		4
	R*n*	CR*m*		1010_*nnnn*_*mmmm*_0110		1
	[EA]	CER*m*		1111_*mmm*0_1010_1101		1
	[EA+]	CER*m*		1111_*mmm*0_1011_1101		1
	[EA]	CR*m*		1111_*mmmm*_1000_1101		1
	[EA+]	CR*m*		1111_*mmmm*_1001_1101		1
	[EA]	CXR*m*		1111_*mm*00_1100_1101		2
	[EA+]	CXR*m*		1111_*mm*00_1101_1101		2
	[EA]	CQR*m*		1111_*m*000_1110_1101		4
	[EA+]	CQR*m*		1111_*m*000_1111_1101		4

Lệnh truyền dữ liệu thanh ghi EA

Mnemonic	First operand	Second operand	Flag changes C Z S OV MIE HC	Instruction code First word	Second	Minimum execution time
LEA	[ER*m*]			1111_0000_*mmm*0_1010		1
	Disp16[ER*m*]			1111_0000_*mmm*0_1011	DDDD_DDDD_DDDD_DDDD	2
	Dadr			1111_0000_0000_1100	DDDD_DDDD_DDDD_DDDD	2

Lệnh ALU

Mnemoni	Firs operand	Secon operand	Flag C Z S OV MIE HC	Instruction code First word	Second	Minimum execution time (cycles)
DAA	R*n*		* * * *	1000_*nnnn*_0001_1111		1
DAS	R*n*		* * * *	1000_*nnnn*_0011_1111		1
NEG	R*n*		* * * * *	1000_*nnnn*_0101_1111		1

Lệnh truy cập Bit

Mnemoni	First operan	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
SB	R*n.bit_offset*			*					1010_*nnnn*_0bbb_0000		1
	Dbitadr			*					1010_0000_1bbb_0000	DDDD_DDDD_DDDD_DDDD	2
RB	R*n. bit_offset*			*					1010_*nnnn*_0bbb_0010		1
	Dbitadr			*					1010_0000_1bbb_0010	DDDD_DDDD_DDDD_DDDD	2
TB	R*n. bit_offset*			*					1010_*nnnn*_0bbb_0001		1
	Dbitadr			*					1010_0000_1bbb_0001	DDDD_DDDD_DDDD_DDDD	2

Lệnh truy cập PSW

Mnemoni	First operan	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
EI							*		1110_1101_0000_1000		1
DI							*		1110_1011_1111_0111		3
SC			*						1110_1101_1000_0000		1
RC			*						1110_1011_0111_1111		1
CPLC			*						1111_1110_1100_1111		1

Lệnh nhánh quan hệ có điều kiện

Mnemoni	First operan	Second operan	Flag changes C	Z	S	OV	MIE	Instruction First	Second word	Minimum time
BGE	*Radr*							1100_0000_rrrr_rrrr		
BLT								1100_0001_rrrr_rrrr		
BGT								1100_0010_rrrr_rrrr		
BLE								1100_0011_rrrr_rrrr		
BGES								1100_0100_rrrr_rrrr		
BLTS								1100_0101_rrrr_rrrr		A34 core: 1/3 (*1)
BGTS								1100_0110_rrrr_rrrr		
BLES								1100_0111_rrrr_rrrr		A35 core 1/2 (*1)
BNE								1100_1000_rrrr_rrrr		
BEQ								1100_1001_rrrr_rrrr		
BNV								1100_1010_rrrr_rrrr		
BOV								1100_1011_rrrr_rrrr		
BPS								1100_1100_rrrr_rrrr		
BNS								1100_1101_rrrr_rrrr		
BAL								1100_1110_rrrr_rrrr		3

*1: Số đếm cao hơn dành cho khi điều kiện phân nhánh được đáp ứng; số đếm thấp hơn dành cho khi điều kiện phân nhánh không được đáp ứng.

Lệnh mở rộng dấu

Mnemoni	First operand	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time (cycles)
EXTBW	ER*n*			*	*				1000_*nnn*1_*nnn*0_1111		

Lệnh ngắt phần mềm

Mnemoni	First operan	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
SWI	*#snum*						*		1110_0101_00ii_iiii		3
BRK									1111_1111_1111_1111		7

Lệnh nhánh

Mnemoni	First operan	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
B	*Cadr*								1111_gggg_0000_0000	cccc_cccc_cccc_cccc	2
	ER*n*								1111_0000_*nnn*0_0010		2
BL	*Cadr*								1111_gggg_0000_0001	cccc_cccc_cccc_cccc	2
	ER*n*								1111_0000_*nnn*0_0011		2

Lệnh nhân và chia

Mnemoni	First operan	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
MUL	ER*n*	R*m*		*					1111_*nnn*0_*mmmm*_0100		9
DIV	ER*n*	R*m*	*	*					1111_*nnn*0_*mmmm*_1001		17

Miscellaneous

Mnemoni	First operan	Second operand	Flag changes C	Z	S	OV	MIE	HC	Instruction code First word	Second	Minimum execution time
INC	[EA]			*	*	*		*	1111_1110_0010_1111		2
DEC	[EA]			*	*	*		*	1111_1110_0011_1111		2
RT									1111_1110_0001_1111		2
RTI			*	*	*	*	*	*	1111_1110_0000_1111		2
NOP									1111_1110_1000_1111		1

LỊCH SỬ SỬA ĐỔI

LỊCH SỬ SỬA ĐỔI

Tài liệu số.	Ngày phát hành	Trang		Mô tả
		Phiên bản trước	Phiên bản hiện tại	
PEUL-U16-100-INST-02	Tháng 7 năm 2013	–	–	Ấn bản lần 2.
PEUL-U16-100-INST-03	Tháng 1 năm 2015	P1-1	P1-1	Đã thêm lời giải thích về loại lõi CPU.
		P3-12 P4-5	P3-12 P4-5	Câu lệnh về số chu kỳ thực hiện của lệnh nhánh có điều kiện được chia cho từng loại CPU.