* Rd: thanh ghi nguồn và cũng là đích thuộc Register File.
* Rr: thanh ghi nguồn thuộc Register File.

Khái niệm nguồn (Source), đích (Destination) là chỉ các toán hạng và kết quả trong các phép toán đại số và Logic

Ví dụ: ADD R1, R2 là lệnh cộng 2 giá trị chứa trong 2 thanh ghi R1, R2, trong trường hợp này cả R1 và R2 đều được gọi là nguồn vì chứa giá trị trước khi thực hiện phép cộng. Sau khi phép cộng được thực hiện, kết quả được chứa lại trong R1 và vì thế R1 được gọi là đích trong trường hợp này. R1 vừa là nguồn, vừa là đích trong khi R2 chỉ là nguồn, nếu viết ví dụ này dưới dạng tổng quát sẽ là : ADD Rd, Rr.

* R: kết quả sau khi lệnh được thực thi.
* K: hằng số.
* k: hằng số chỉ địa chỉ tuyệt đối của thanh ghi.
* b: (0 đến 7) số thứ tự bit trong các thanh ghi của Register File và vùng nhớ I/O.
* s: (0 đến 7) số thứ tự bit trong thanh ghi trạng thái SREG.
* X,Y,X: các thanh ghi địa chỉ tương đối (X=R27:R26, X=R29:R28, X=R31:R30).
* A: địa chỉ I/O.
* q: độ dịch chuyển của địa chỉ tuyệt đối.

**I. Instruction chỉ dùng cho Register Files.**

- **LDI** (LoaD Immediate).

* Cú pháp: LDI Rd,K
* Chức năng: Load hằng số K vào thanh ghi Rd.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ: LDI R16, 99 kết quả là thanh ghi R1 mang giá trị 99.

- **MOV** (MOVE).

* Cú pháp: MOV Rd, Rr
* Chức năng: Copy giá trị trong thanh ghi Rr vào thanh ghi Rd.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ: MOV R15, R16 kết quả là R15 có cùng giá trị với R16 (R15=R16=99).

- **CLR** (CLEAR Register).

* Cú pháp: CLR Rd
* Chức năng: Copy giá trị trong thanh ghi Rr vào thanh ghi Rd.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ: áp dụng cho tất cả các thanh ghi trong RF.

- **SER** (SET Register).

* Cú pháp: SER Rd
* Chức năng: set tất cả các bit tronh thanh ghi Rd lên 1, sau lệnh này thanh ghi Rd=0xFF.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ: SER R16 kết quả là R16 = 0xFF.

- **CBR** (CLEAR Bit in Register).

* Cú pháp: CBR Rd, K
* Chức năng: xóa các bit trong thanh ghi Rd với “mặt nạ”  K, nếu Bit nào trong K là 1 thì Bit tương ứng trong Rd sẽ bị xóa.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ: CBR R16, 0xF0  kết quả là 4 bit cao nhất của  R16 bị xóa vì K=11110000 (B).

- **SBR** (SET Bit in Register).

* Cú pháp: SBR Rd, K
* Chức năng: set các bit trong thanh ghi Rd với “mặt nạ”  K, nếu Bit nào trong K là 1 thì Bit tương ứng trong Rd sẽ được set lên 1.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ: SBR R16, 0xF0  kết quả là 4 bit cao nhất của  R16 được set lên 1 vì K=11110000 (B).

- **BLD** (Bit LoaD from T Flag).

* Cú pháp: BLD Rd,b
* Chức năng: Load giá trị trong cờ T của thanh ghi SREG vào bit thứ b trong thanh ghi Rd. Đây cũng chính là chức năng chính của cờ T.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  SET ; set bit T lên 1  
  BLD R16, 4  
  Kết quả là bit 4 của thanh ghi R16 được set lên 1 vì giá trị của bit T là 1.

- **BST** (Bit Storage from T Flag).

* Cú pháp: BST Rd,b
* Chức năng: Copy bit thứ b trong thanh ghi Rd vào trong cờ T của thanh ghi SREG. Đây cũng chính là chức năng chính của cờ T.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ: BST R16, 4 kết quả là cờ T chứa giá trị của bit 4 của thanh ghi R16.

- **CPI** (COMPARE with Immediate).

* Cú pháp: CPI Rd, K
* Chức năng: so sánh thanh ghi Rd với hằng số K, lệnh này làm thay đổi nhiều bit trong thanh ghi SREG trong đó sự thay đổi của cờ Zero là quan trọng nhất, nếu Rd = K cờ Z=1, ngược lại Z=0, sử dụng đặc điểm thay đổi của cờ Z kết hợp với lệnh BRNE hoặc BREQ chúng ta có thể tạo thành một lệnh rẽ nhánh.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ:  
  LDI R16, 10  
  CPI R16, 10  
  Kết quả là cờ Z được set thành 1 vì lúc này R16 =10.

- **ANDI** (AND with Immediate).

* Cú pháp: ANDI Rd, K
* Chức năng: thực hiện phép Logic AND giữa thanh ghi Rd với hằng số K và kết quả đặt lại trong Rd.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ: ANDI R17, 0x00 kết quả là R17 có 0x00.

- **AND** (Logical AND).

* Cú pháp: AND Rd, Rr
* Chức năng: thực hiện phép Logic AND giữa 2 thanh ghi Rd và Rr , kết quả đặt lại trong Rd.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  LDI R1, 0xFF ;(11111111)  
  LDI R17, 0xAA; (10101010)  
  AND R1, R17  
  Kết quả là R1=0xAA vì 11111111 & 10101010 =10101010.

- **ORI** (Logical OR with Immediate).

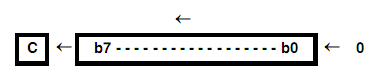
* Cú pháp: ORI Rd, K
* Chức năng: thực hiện phép Logic OR giữa thanh ghi Rd với hằng số K và kết quả đặt lại trong Rd.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ: ORI R17, 0xFF kết quả là R17 có 0xFF.

- **OR** (Logical OR).

* Cú pháp: OR Rd, Rr
* Chức năng: thực hiện phép Logic OR giữa 2 thanh ghi Rd và Rr , kết quả đặt lại trong Rd.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  LDI R1, 0xFF ;(11111111)  
  LDI R17, 0xAA; (10101010)  
  OR R1, R17  
  Kết quả là R1=0xFF vì 11111111 or 10101010 =11111111.

- **LSL** (Logical Shift Left).

* Cú pháp: LSL Rd
* Chức năng: dịch tất thanh ghi Rd sang trái 1 vị trí, Bit 7 (bit lớn nhất) của Rd sẽ được chứa trong cờ nhớ C, bit 0 của Rd bị xóa thành 0. Thực chất LSL tương đương với phép nhân thanh ghi Rd với 2. Bạn xem hình minh họa bên dưới.

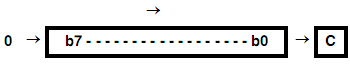
[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEgmJLEQmhnnBm3vCe_wmUaNCrHF9miMu-Uh6dbFa4z0F7dBKnAAXXdiV_ZsoBZmj30thGDqyvRKwDUHXbXkXBCVZZArWjstSdWTqQ_3jyi6FLmuDggEHYQPT9o2ORrbbMqQ3oPNwMybT-T3/s1600/image017.png)

* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  LDI R1, 0B11000011 ; (dạng nhị phân của 195)  
  LSL R1

Kết quả là R1=10000110 và cờ C =1 vì thanh ghi R1 đã được dịch sang trái 1 vị trí, trước khi dịch bit 7 của R1 là 1 nên sau khi dịch bit này được chứa trong C, cho nên C=1.

- **LSR** (Logical Shift Right).

* Cú pháp: LSR Rd
* Chức năng: dịch tất thanh ghi Rd sang phải 1 vị trí, Bit 0 (bit nhỏ nhất) của Rd sẽ được chứa trong cờ nhớ C, bit 7 của Rd bị xóa thành 0. Thực chất LSR tương đương với phép chia thanh ghi Rd cho 2. Bạn xem hình minh họa bên dưới.

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEhgweFiC9ySBlazUcTVm95OCoVFiw4OG0pmdL53ZopF52gdbzquu0-NwV0lLCKAvHLzw53KS9vzHWDjFT44an-kBAx6IXDgseNykEEp_sVMs_a85kRLVdpg1UpbmAzVLEnqQmUC-Z01Ofh6/s1600/image019.png)

* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  LDI R1, 0B11000110 ; (dạng nhị phân của 195)  
  LSR R1

Kết quả là R1=01100001 và cờ C =1 vì thanh ghi R1 đã được dịch sang phải 1 vị trí, trước khi dịch bit 0 của R1 là 1 nên sau khi dịch bit này được chứa trong C, cho nên C=1.

- **ADD** (ADD without Carry).

* Cú pháp: ADD Rd, Rr
* Chức năng: thực hiện phép cộng 2 thanh ghi Rd và Rr , kết quả đặt lại trong Rd. Cờ nhớ C không được sử dụng.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  LDI R16, 30  
  LDI R17, 25  
  ADD R16, R17  
  Kết quả là R16=55.

- **INC** (INCrement).

* Cú pháp: INC Rd
* Chức năng: tăng thanh ghi Rd 1 đơn vị và kết quả đặt lại trong Rd. Lệnh này đặc biệt thích hợp cho các ứng dụng lặp, kết hợp với BREQ hay BRNE có thể tạo thành 1 vòng lặp FOR.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ: INC R17 kết quả là R17 được tăng thêm 1 đơn vị.

- **SUB** (SUBtract without Carry).

* Cú pháp:  **SUB** Rd, Rr
* Chức năng: thực hiện phép trừ 2 thanh ghi Rd - Rr , kết quả đặt lại trong Rd. Cờ nhớ C không được sử dụng.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  LDI R16, 30  
  LDI R17, 25  
  SUB R16, R17  
  Kết quả là R16=5.

- **SUBI** (SUBtract Immediate).

* Cú pháp:  **SUBI** Rd, K
* Chức năng: thực hiện phép trừ  thanh ghi Rd với hằng số K, kết quả đặt lại trong Rd.
* Giới hạn: chỉ áp dụng cho các thanh ghi từ R16 đến R31.
* Ví dụ:  
  LDI R16, 30  
  SUBI R16, 20  
  Kết quả là R16=10.

- **DEC** (DECrement).

* Cú pháp: DEC Rd
* Chức năng: giảm thanh ghi Rd 1 đơn vị và kết quả đặt lại trong Rd. Lệnh này đặc biệt thích hợp cho các ứng dụng lặp, kết hợp với BREQ hay BRNE có thể tạo thành 1 vòng lặp FOR.
* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ: DEC R17 kết quả là R17 được giảm đi 1 đơn vị.

- **MUL** (MULtiply unsigned).

* Cú pháp: MUL Rd, Rr
* Chức năng: thực hiện phép nhân không dấu 2 thanh ghi 8 bit Rd, Rr, kết quả là 1 số 16 bit đặt trong 2 thanh ghi R1:R0. Chú ý nếu Rd và Rr là các thanh ghi R1 và R0 thì kết quả sau khi tính được sẽ được viết đè lên. Xem hình minh họa instruction MUL bên dưới.

[A black rectangle with a blue arrow next to it

Description automatically generated](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEg553JKsySi_aqRiaJWasc2TvNqd8bWFknbGABq9Bk7RHildQzBUB0mglTEcsEPXDvn9ZRclZCJpQPBVcUIts-wiZMEhc4NBfPsac2jtSAJZ3QQiVZu6FUOe73LmsjxKU-lAsAaRUAEEsGO/s1600/image021.png)

* Giới hạn: áp dụng cho tất cả các thanh ghi trong RF.
* Ví dụ:  
  LDI R16, 30  
  LDI R17, 25  
  MUL R16, R17  
  Kết quả là R1=0x2, R0=0xEE, vì 30x25=750=0x02EE.

**II. Instruction cho các thanh ghi I/O.**

Bốn instruction sau đây được thiết kế riêng để truy cập vùng nhớ I/O, các instruction này sử dụng địa chỉ I/O của các thanh ghi trong vùng nhớ này. Vì là thiết kế riêng cho vùng nhớ I/O, bạn không thể sử dụng các thanh ghi này để truy cập RF hay SRAM. Trong các cú pháp của instruction này, khái niệm địa chỉ A là địa chỉ I/O, 0 ≤ A ≤ 63, nếu trong ví dụ A=0x00 thì đó là thanh ghi đầu tiên của vùng I/O, không phải là thanh ghi R0.

- **OUT** (OUTPUT Data).

Cú pháp: OUT A, Rr

Chức năng: xuất giá trị từ thanh ghi Rr ra thanh ghi có địa chỉ A trong vùng nhớ I/O. đây là cách phổ biến nhất để xuất giá trị ra vùng I/O.

Giới hạn: Rr là thanh ghi RF bất kỳ, A bị giới hạn từ 0 đến 63.

Ví dụ:

LDI R16, 0xFF

OUT 0x11, R16

Kết quả là thanh ghi có địa chỉ 0x11 trong vùng I/O, tức thanh ghi DDRD, có giá trị bằng 0xFF.

- **IN** (INPUT Data).

Cú pháp: IN Rr, A

Chức năng: Load giá trị từ thanh ghi có địa chỉ A trong vùng nhớ I/O vào thanh ghi Rr. Đây là cách phổ biến nhất để nhận giá trị từ vùng I/O.

Giới hạn: Rr là thanh ghi RF bất kỳ, A bị giới hạn từ 0 đến 63.

Ví dụ:

IN R16, 0x10

Kết quả là thanh ghi R16 nhận được giá trị của thanh ghi có địa chỉ 0x11 trong vùng I/O, tức thanh ghi PIND, đây chính là ví dụ đọc giá trị các chân của PORTD vào R16.

- **SBI** (Set Bit in I/O Register).

Cú pháp: SBI A, b

Chức năng: Set bit thứ b trong thanh ghi có địa chỉ A trong vùng nhớ I/O. Tuy nhiên lệnh này không có tác dụng trên toàn bộ vùng I/O mà chỉ có tác đối với 32 thanh ghi đầu (địa chỉ từ 0 đến 31).

Giới hạn: b là số thứ các bit trong thanh ghi, 0≤b≤7; A bị giới hạn từ 0 đến 31.

Ví dụ:

SBI 0x12, 2

Kết quả là bit 2 của thanh ghi có địa chỉ 0x12 trong vùng I/O, tức thanh ghi PORTD, được set lên 1. Đây chính là ví dụ set chân PD2 của PORTD.

- **CBI** (Clear Bit in I/O Register).

Cú pháp: CBI A, b

Chức năng: xóa bit thứ b trong thanh ghi có địa chỉ A trong vùng nhớ I/O. Tuy nhiên lệnh này không có tác dụng trên toàn bộ vùng I/O mà chỉ có tác đối với 32 thanh ghi đầu (địa chỉ từ 0 đến 31).

Giới hạn: b là số thứ các bit trong thanh ghi, 0≤b≤7; A bị giới hạn từ 0 đến 31.

Ví dụ:

CBI 0x12, 2

Kết quả là bit 2 của thanh ghi có địa chỉ 0x12 trong vùng I/O, tức thanh ghi PORTD, bị xóa thành 0. Đây chính là ví dụ xóa chân PB2 của PORTD.

**III. Các con trỏ X, Y, Z và cách truy cập toàn bộ không gian bộ nhớ.**

Trong Register File của AVR, các thanh ghi từ R26 đến R31ngoài chứa năng thanh ghi thông thường còn có chức năng là con trỏ (Pointer) trong việc truy cập bộ nhớ (cả bộ nhớ data và bộ nhớ Program). Nếu được sử dụng như các Pointer, các thanh ghi trên được biết đến với tên gọi X, Y, Z. Định nghĩa như sau: X=R27:R26, Y=R29:R28, Z=R31:R30. Chúng là 3 thanh ghi 16 bit được định nghĩa trước cho tất cả các AVR. Ngoài ra trong các file định nghĩa cho chip chúng ta có thêm 6 định nghĩa khác là XL, XH, YL, YH, ZL, ZH cũng chính là tên gọi của R26-> R31. Phần này chúng ta khảo sát một số instruction dùng truy cập toàn bộ khồi nhớ của AVR bằng cách sử dụng địa chỉ trực tiếp và bằng cách sử dụng Pointer.

- **LDS** (LoaD direct from data Space).

Cú pháp: LDS Rd, k

Chức năng: load giá trị 1 byte từ thanh ghi có địa chỉ k trong SRAM vào thanh ghi Rd, k là dạng địa chỉ tuyệt đối có giới hạn từ 0 đến 65535(2^16-1).

Giới hạn: Rd là thanh ghi bất kỳ trong RF nhưng giá trị lớn nhất của k là 65535, vì thế với lệnh này ta không thể truy cập vượt quá khoảng không gian 64KB. Nếu muốn truy cập vùng không gian lớn hơn 64KB chúng ta cần một số hỗ trợ, tuy nhiên ở đây tôi giả sử bộ nhớ của chip (thường là bộ nhớ data) không vượt quá 64KB (thực tế chưa có chip AVR nào có SRAM hay EEPROM vượt quá 64KB).

Ví dụ:

LDS R2, 0x0060

Kết quả là thanh ghi R2 chứa giá trị của thanh ghi có địa chỉ 0x0060, đây là thanh ghi đầu tiên trong khoảng SRAM (sau RF và vùng I/O) của AVR.

- **STS** (STorage direc to data Space).

Cú pháp: STS k, Rr

Chức năng: instruction này hoàn toàn giống LDS nhưng dùng để xuất dữ liệu từ thanh ghi Rr ra RAM, ngươi đọc có thể tham khảo phần giải thích cho LDS.

Sử dụng địa chỉ trực tiếp thì câu lệnh sẽ đơn giản nhưng rất khó nhớ phần địa chỉ, thông thường SRAM là vùng chúng ta hay sử dụng để chứa biến tạm thời, trong các ngôn ngữ cấp cao ta chỉ cần nhớ tên biến nhưng với ASM chúng ta phải nhớ địa chỉ của chúng. Một cách tốt để tránh việc này là dùng chỉ thị (DIRECTIVE, bạn xem lại bài 1) . EQU để gán tên biến cho 1 địa chỉ, ví dụ .EQU bientam = 0x0060 và sau đó sử dụng bientam thay cho 0x0060.

Một cách khác được dùng để truy cập bộ nhớ mà không dùng địa chỉ tuyệt đối là sử dụng sử dụng con trỏ. Có 2 instruction hỗ trợ con trỏ là LD(LoaD indirec from data Space), và ST (STorage indirec to data Space), LD đọc dữ liệu từ SRAM vào thanh ghi còn ST lưu dữ liệu từ thanh ghi vào SRAM. Cả 3 con trỏ X, Y và Z đều có thể được dùng nhưng có một số điểm lưu ý: cả 3 đều dùng được trong trường hợp truy xuất thông thường nhưng với cách truy cập có offset, con trỏ X không sử dụng được. Để truy xuất bộ nhớ chương trình bằng con trỏ thì Z là giải pháp duy nhất…Dưới đây là 1 số cách sử dụng LD, ST kết hợp với con trỏ, chúng ta xét thông qua các ví dụ.

**IV. Rẽ nhánh và vòng lặp.**

Bên cạnh BRNE chúng ta có 1 số instruction phục vụ rẽ nhánh khác như:

- **BREQ** (BRanch if EQual).

Cú pháp: BREQ LABEL

Chức năng: Nhảy đến nhãn LABEL nếu cờ Z =1. Cờ Z chịu tác động của rất nhiều instruction như CP, CPI, SUB, SUBI…vì thế BREQ thường được sử dụng sau các instruction này.

Ví dụ:

LDI R16, 0xFF

LDI R17, 0xFF

CP R16, R17 ; so sanh 2 thanh ghi R16, R17

BREQ RENHANH

…..

RENHANH:

; thực hiện những việc khi rẽ nhánh.

Kết quả là việc rẽ nhánh xảy ra vì khi so sánh bằng CP, R17=R16 nên cờ Z tự động được set bằng 1, lệnh BREQ được thực thi và nhảy đến nhãn RENHANH. Ví dụ này tương đương cấu trúc if (R16=R17) {thực hiện những việc khi rẽ nhánh}.

- **BRLO** (BRanch if LOwer).

Cú pháp: BRLO LABEL

Chức năng: bản chất của câu lệnh là nhảy đến nhãn LABEL nếu cờ C =1. Tuy nhiên, thông thường lệnh này sử dụng theo sau các instruction như CP, CPI, SUB, SUBI…khi đó việc rẽ nhánh sẽ xảy ra nếu thanh ghi Rd

Ví dụ:

EOR R16, R16 ;XOR R16 với chính nó, tương đương CLR R16

VONG LAP:

INC R16 ;tăng R16 thêm 1 đơn vị

CPI R16, $10 ;so sánh R16 với số hexadecimal $10

BRLO VONGLAP ;nhảy về VONGLAP nếu R16 <$10

NOP ;câu lệnh này sẽ được thực thi nếu điều kiện rẽ nhánh ở trên không thỏa,

; NOP là 1 instruction, chức năng là không làm gì cả.

Kết quả là phần lệnh bên trong VONGLAP sẽ được thưc hiện khoảng 16 lần ($10=16) trước khi thực hiện lệnh NOP.

- **BRSH** (BRanch if Same or Higher).

Cú pháp: BRSH LABEL

Chức năng: bản chất của câu lệnh là nhảy đến nhãn LABEL nếu cờ C =0. Tuy nhiên, thông thường lệnh này sử dụng theo sau các instruction như CP, CPI, SUB, SUBI…khi đó việc rẽ nhánh sẽ xảy ra nếu thanh ghi Rd ≥Rr.

Ví dụ:

SUBI R16, 4 ;trừ R16 đi 4 đơn vị

BRSH RENHANH ; nhảy đến RENHANH nếu R16 ≥ 4

….

RENHANH:

NOP

…