Atmel AVR CPU Core

1. Tổng quan về kiến trúc:

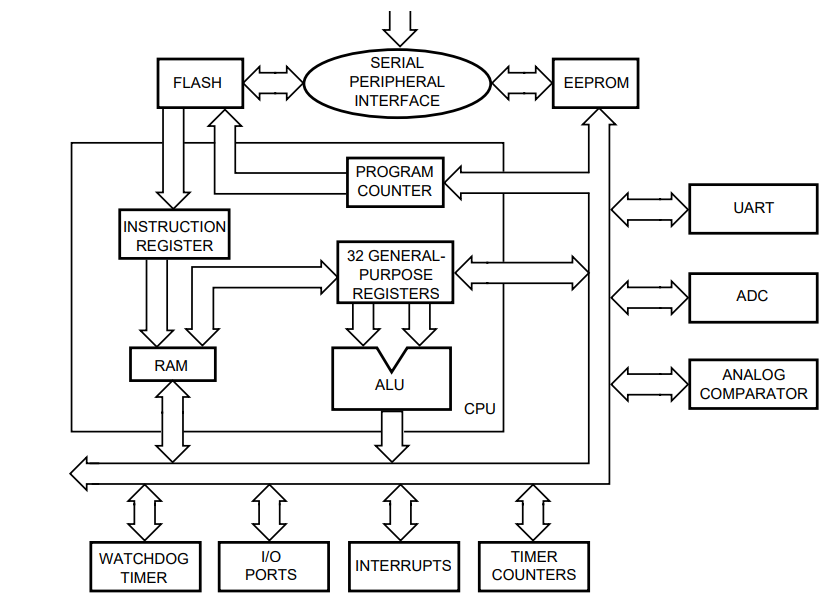
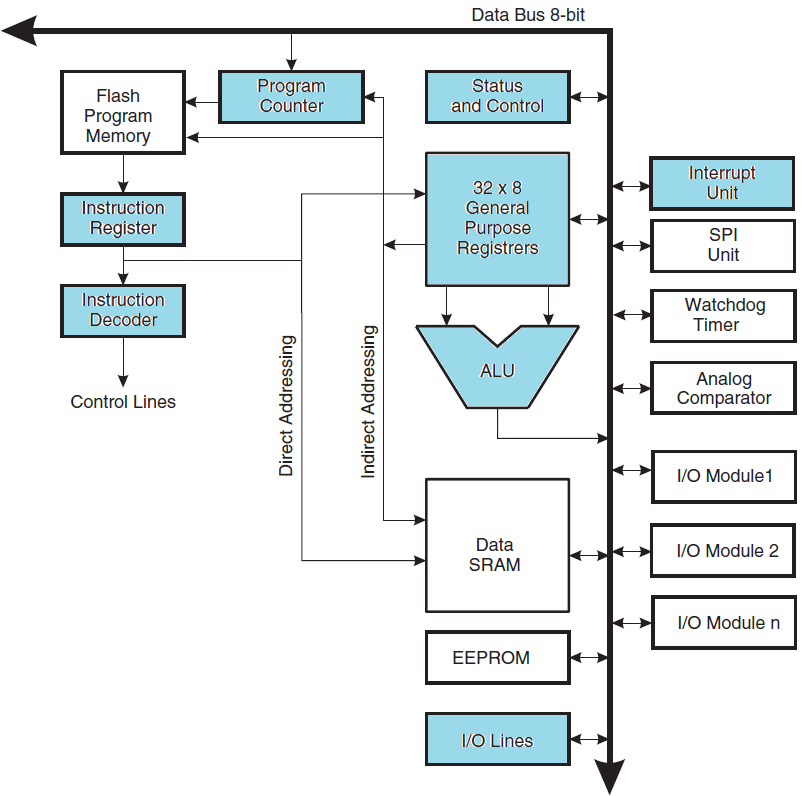


Figure https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=43fb54acc0ef15742845747392176dee3af6662e

* Harvard architecture
* Bộ nhớ chương trình là bộ nhớ Flash
* 6 trong 32 thanh ghi (General purpose registers) có thể được dùng như thanh ghi 16-bit. Giúp lưu trữ dữ liệu của các bit tràn trong phép tính toán. Gồm: X, Y, Z-register
* ALU; Status register is updated to respond the result
* Bộ nhớ của AVR là tuyến tính và có cách sắp xếp thông thường
* Module ngắt sẽ điều khiển hoạt động ngắt ở ngoại vi và hoạt động cho phép ngắt ở Status register. Địa chỉ Vector ngắt càng thấp thì mức độ ưu tiên càng cao.

Instruction set: AVR hỗ trợ khoảng 120 lệnh. Bao gồm các lệnh như: tính toán số học và logic, thay đổi hướng của chương trình, chuyển đổi dữ liệu (data transfer), các lệnh thực thi trên bit,…

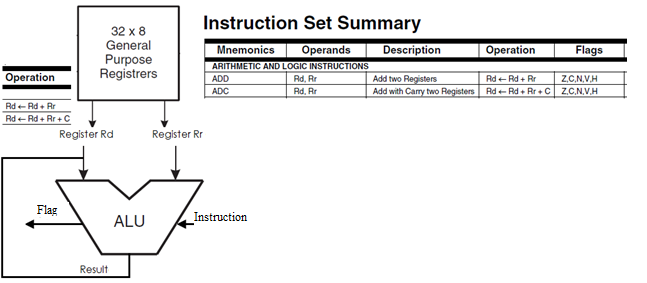
Development tool: AVR studio



1. **ALU - Arithmetic Logic Unit**

ALU là đơn vị trung tâm của bộ xử lý CPU. Các hoạt động chính của ALU là: tính toán số học, xử lý logic, xử lý bit.

ALU của AVR được kết nối trực tiếp với 32 thanh ghi đa dụng giúp làm tăng hiệu năng và tính linh hoạt của quá trình xử lý.



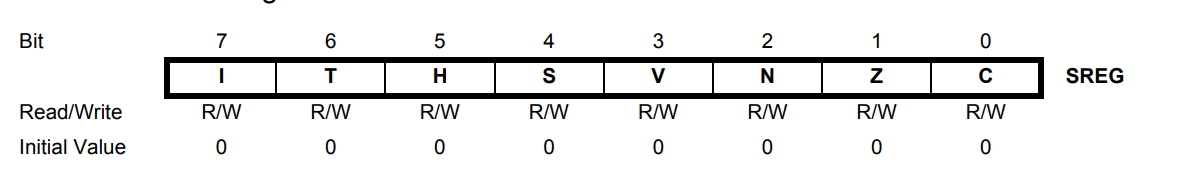
Ảnh : Mô tả hoạt động của ALU

ALU có thể thực hiện các chức năng giữa hai thanh ghi hoặc giữa một hằng số và một thanh ghi. Các phép xử lý tính toán, sau khi được thực thi, sẽ được lưu vào thanh ghi đích Rd. Thanh ghi trạng thái (status register) cũng được cập nhật. Các bit trạng thái trong status register được gọi là Flag (cờ), những flag này có chức năng cung cấp thêm thông tin về hoạt động của ALU. Chi tiết về status register sẽ được đề cập ở phần sau.

Một phép toán số học được thực thi ứng với một xung clock duy nhất.

1. **Status register**

Như đã nói ở phần trên, status register sẽ được cập nhập trạng thái sau mỗi thao tác lệnh. Trạng thái này có thể được sử dụng để thay đổi luồng chương trình nhằm thực hiện các thao tác có điều kiện.



Bit 7: Global interupt Enable(I)

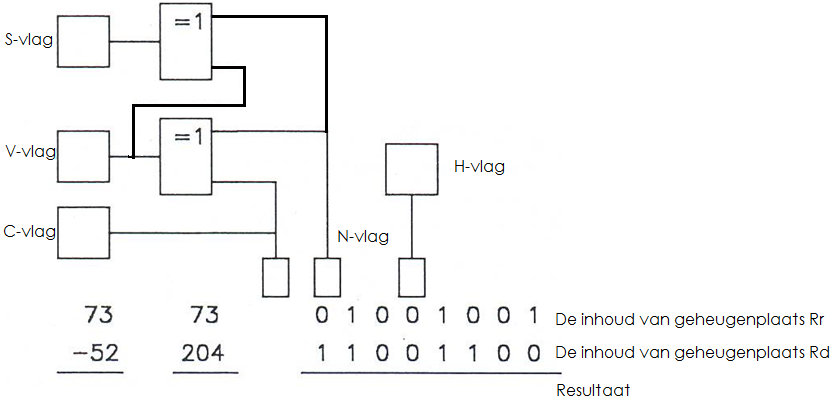
* RETI = Interrupt return (PC ← STACK)
* SEI = Global interrupt Enable
* CLI = Global interrupt Disable

Bit 6: Bit Copy Storage (T)

* BLD = Bit Load
* BST = Bit Store

Bit 5: Half Carry Flag (H)

Bit 4: Sign Bit (S)



Bit 3: Two Complement overflow flag (V)

Bit 2: Negative flag (N)

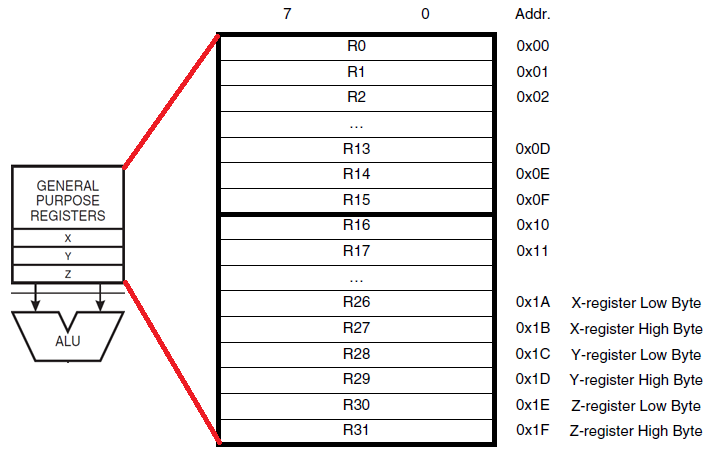
Bit 1: Zero flag (Z)

Bit 0: Carry flag (C)

Lưu ý:

Khi xảy ra ngắt, thanh ghi trạng thái không tự động được lưu ở bất kỳ đâu. Ngoài ra, khi chương trình được tiếp tục sau một chương trình ngắt, thanh ghi trạng thái (như trước chương trình con) sẽ không tự động được gọi lại. Người dùng phải tự lập trình!

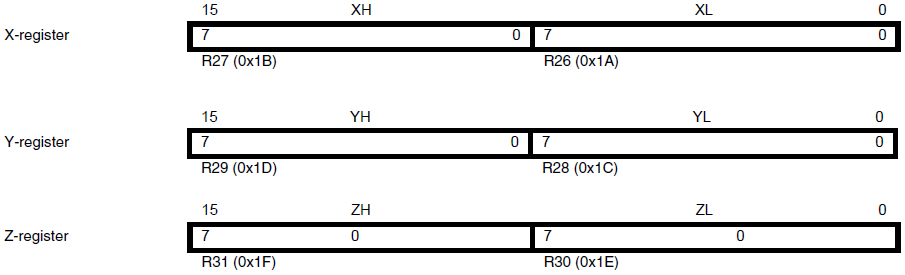
1. **General Purpose Register File**



Note:

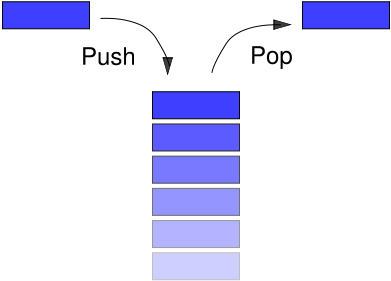
* 32 thanh ghi được gán một ví trí trong bộ nhớ dữ liệu
* 32 địa chỉ đầu tiên của bộ nhớ dữ liệu được dành riêng cho 32 thanh ghi này. Tuy nhiên, nó không được triển khai như một SRAM

1. **The X-register, Yregister and Z-register**



1. **Stack and stack pointer**

Stack là một vùng bộ nhớ được sử dụng để lưu trữ dữ liệu tạm thời, biến cục bộ và trả về địa chỉ sau khi ngắt và sau một chương trình con. Stack được xây dựng dưới dạng thanh ghi LIFO (Last in, first out), tức là các mục được thêm vào cuối cùng sẽ được lấy ra trước. Trong kiến trúc AVR, chúng ta có thể khai báo một vùng nhớ trong SRAM là stack.



Để khai báo một vùng SRAM làm stack chúng ta cần xác lập địa chỉ đầu của stack bằng cách xác lập con trỏ stack pointer (SP). Stack pointer là một con trỏ 16 bit gồm 2 thanh ghi 8 bit SPL và SPH.