EE-222: Microprocessor Systems

AVR Microcontroller: Architecture and Assembly Language

Instructor: Dr. Arbab Latif

NUST

SCHOOL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE (SEECS)

ATmega16A

ATmega16A

Name	Value
Program Memory Type	Flash
Program Memory Size (KB)	16
CPU Speed (MIPS/DMIPS)	16
SRAM Bytes	1,024
Data EEPROM/HEF (bytes)	512
Digital Communication Peripherals	1-UART, 1-SPI, 1-I2C
Capture/Compare/PWM Peripherals	1 Input Capture, 1 CCP, 4PWM
Timers	2 x 8-bit, 1 x 16-bit
Number of Comparators	1
Temperature Range (C)	-40 to 85
Operating Voltage Range (V)	2.7 to 5.5
Pin Count	44

ATmega16A: Features

- Low-power Atmel AVR 8-bit Microcontroller

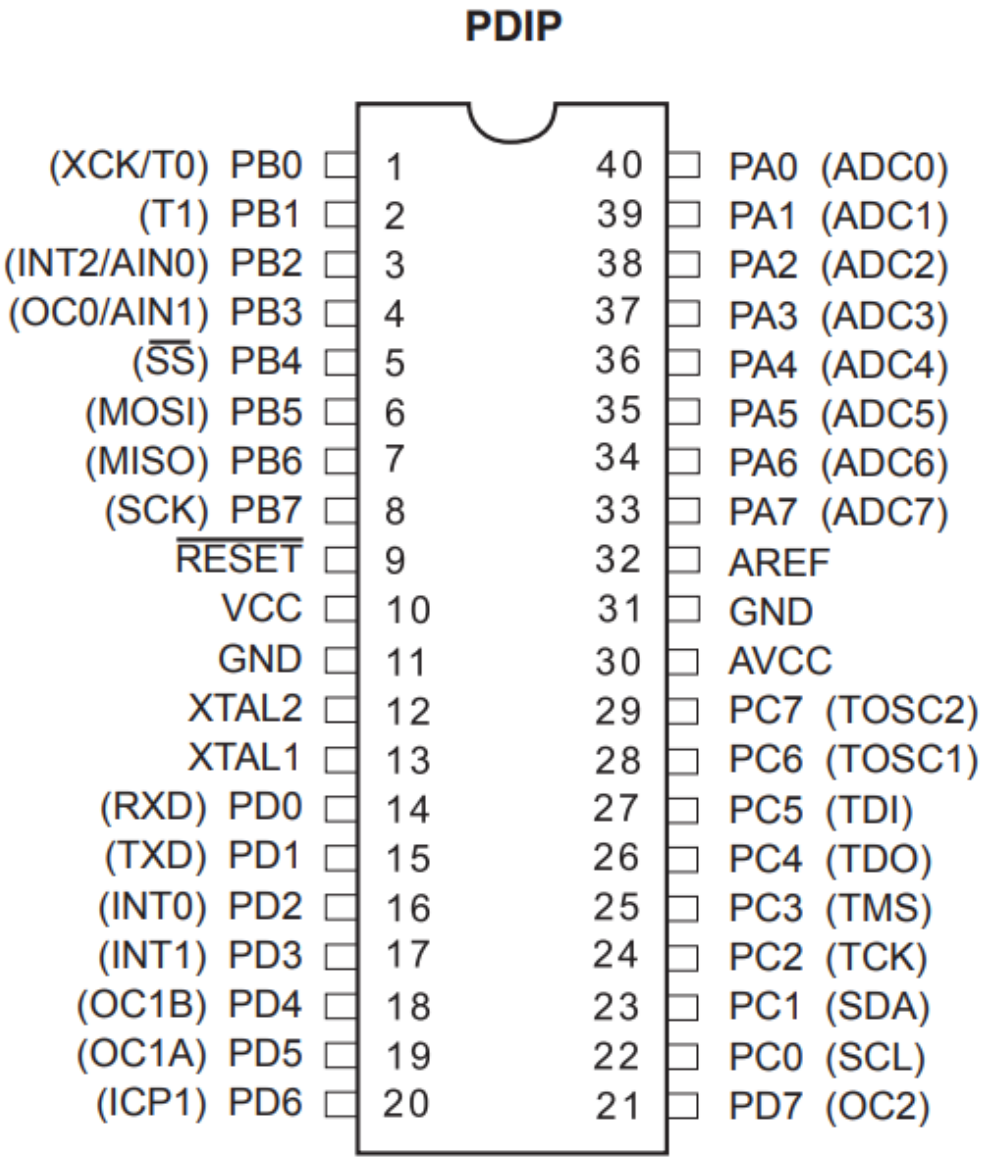
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers

- Peripheral Features
  - Two 8-bit Timer/Counters
  - One 16-bit Timer/Counter Capture Mode
  - 8-channel, 10-bit ADC

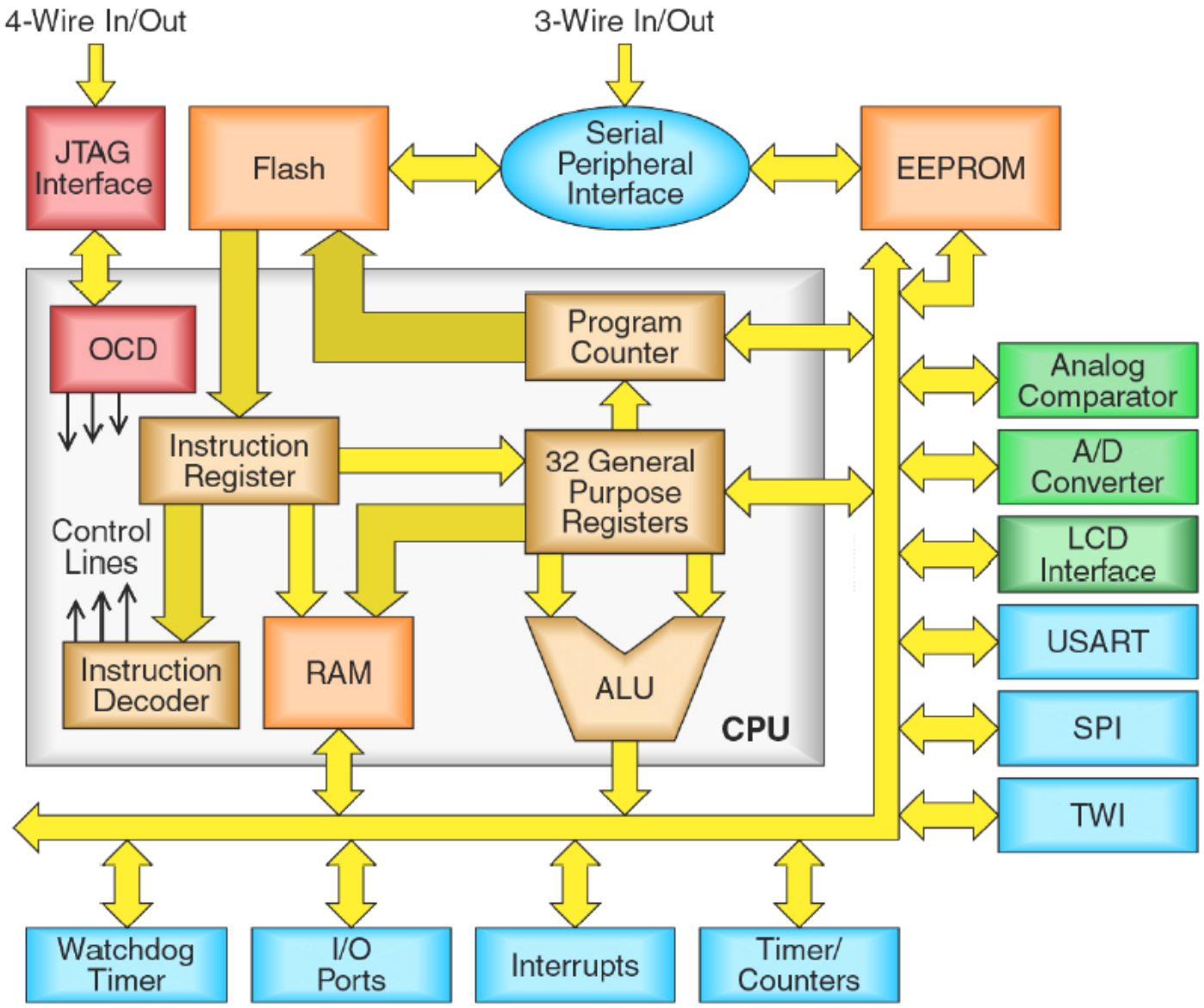
- For more details, take a look at:

http://ww1.microchip.com/downloads/en/devicedoc/atmel-8154-8-bit-avr-atmega16a_datasheet.pdf

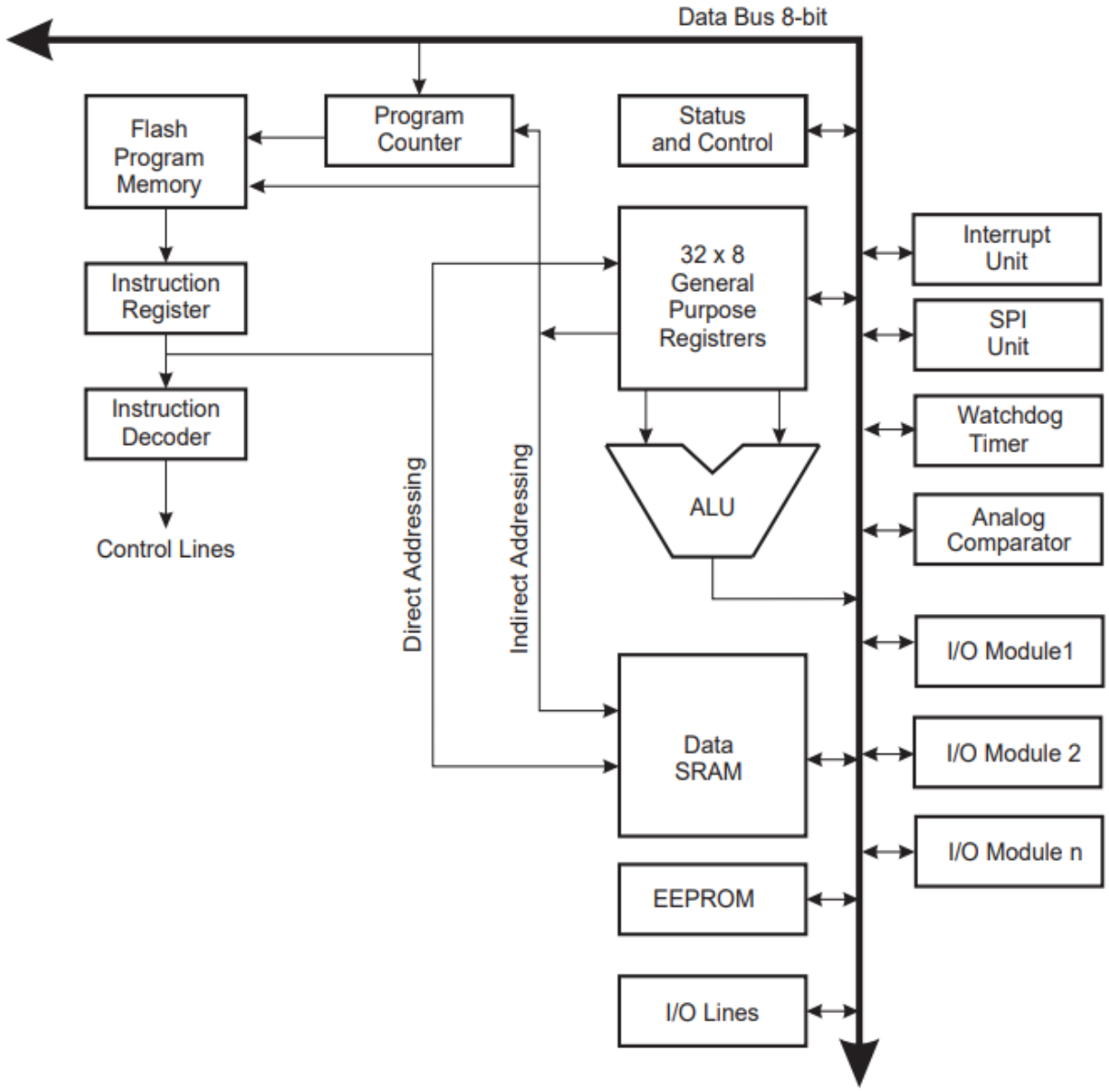
AVR Architecture: Pin-out ATmega16A



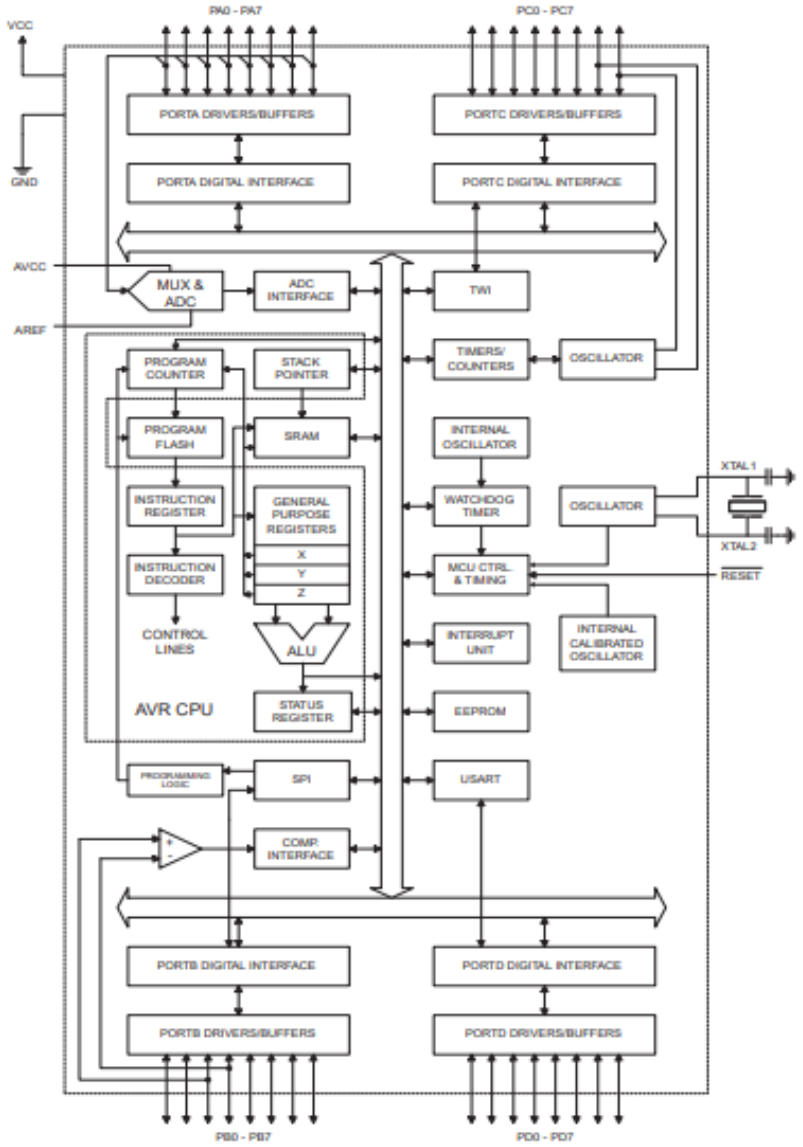
AVR Architecture: Block Diagram



AVR Architecture: Block Diagram (another view)



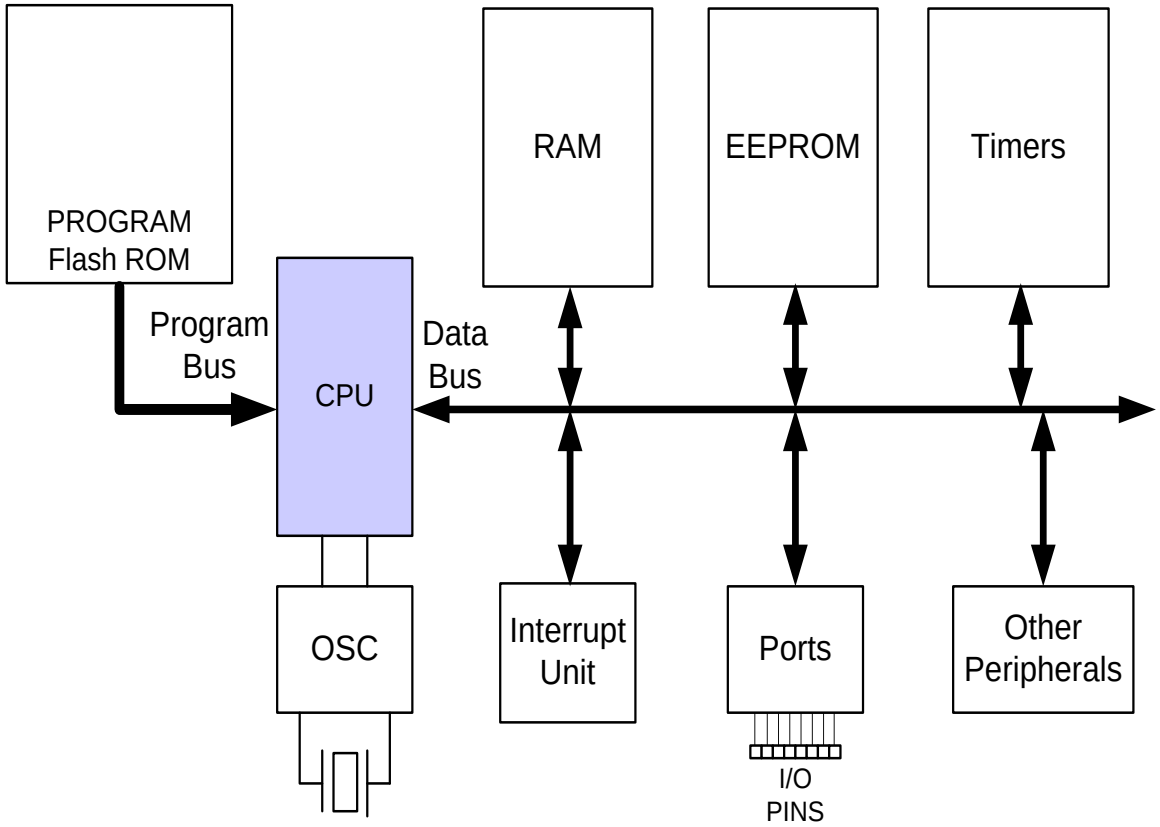
AVR Architecture: Block Diagram (yet another view)



AVR Assembly Language

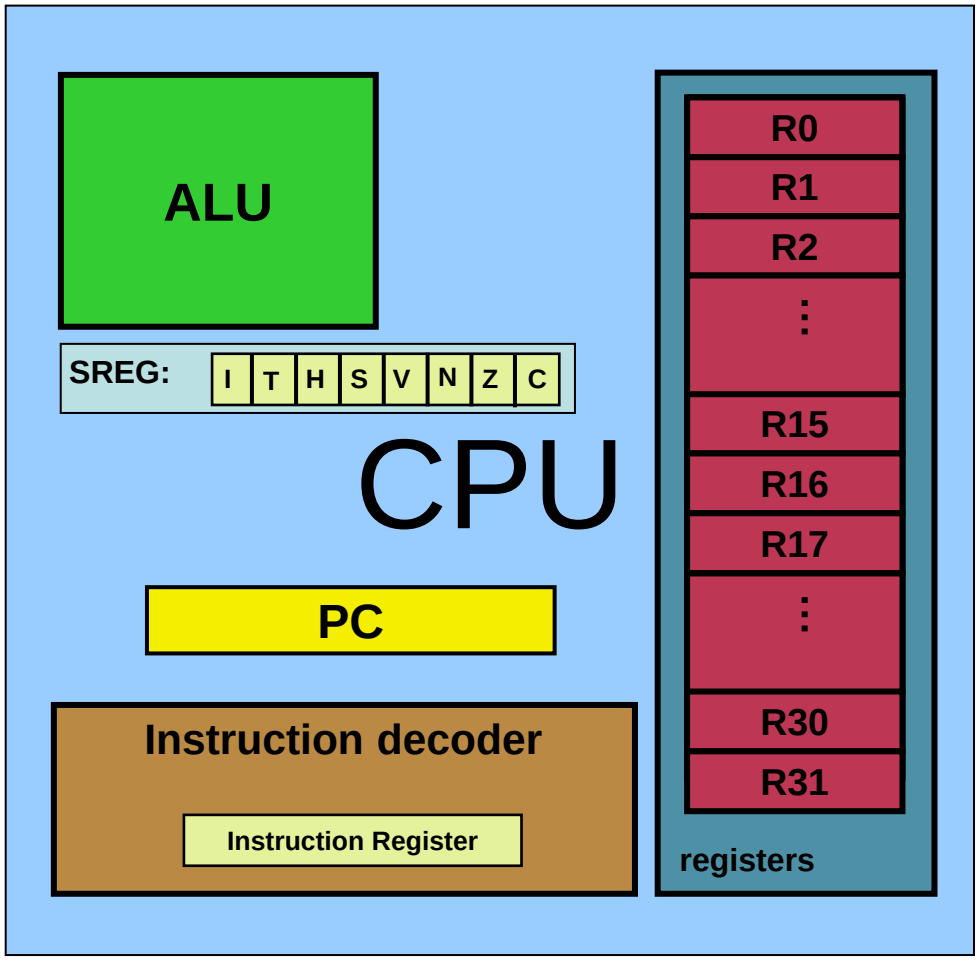
Programming AVR

- To program in Assembly language, we must understand the registers and architecture of a given CPU and the role they play in processing data.



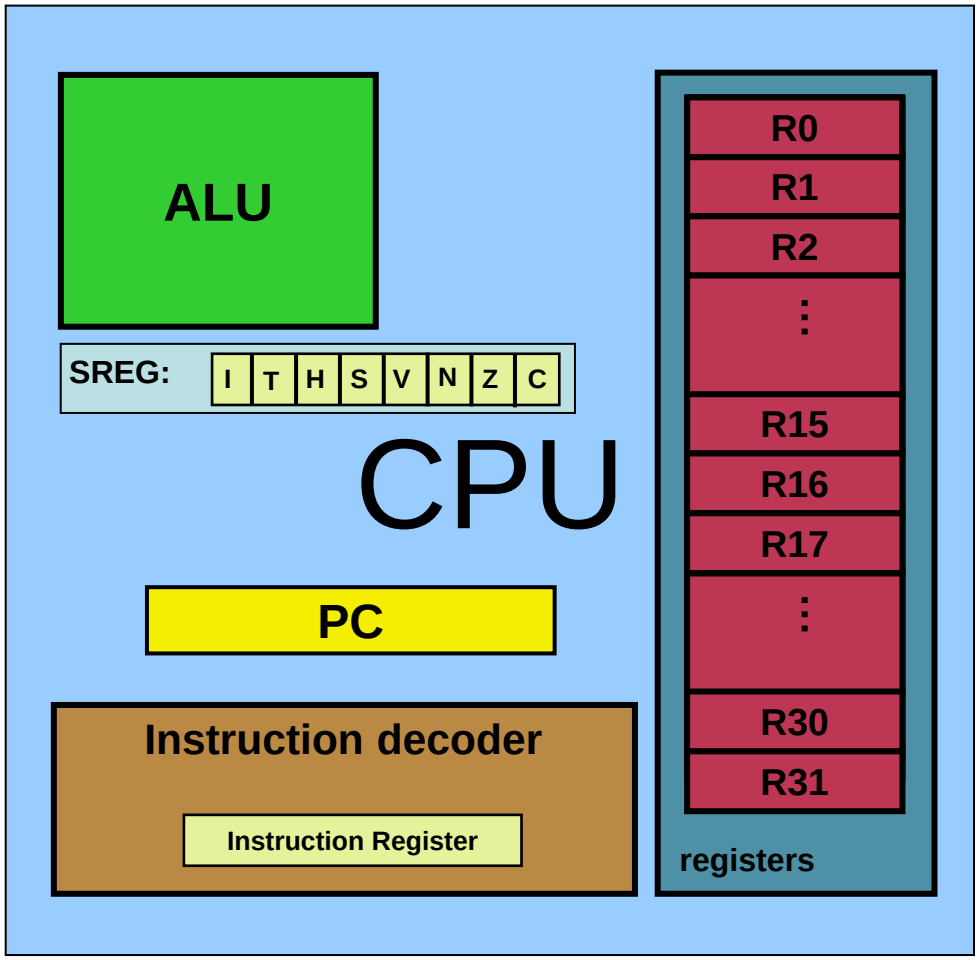
AVR's CPU

- AVR's CPU
  - ALU
  - 32 General Purpose registers (R0 to R31)
  - PC register
  - Instruction decoder



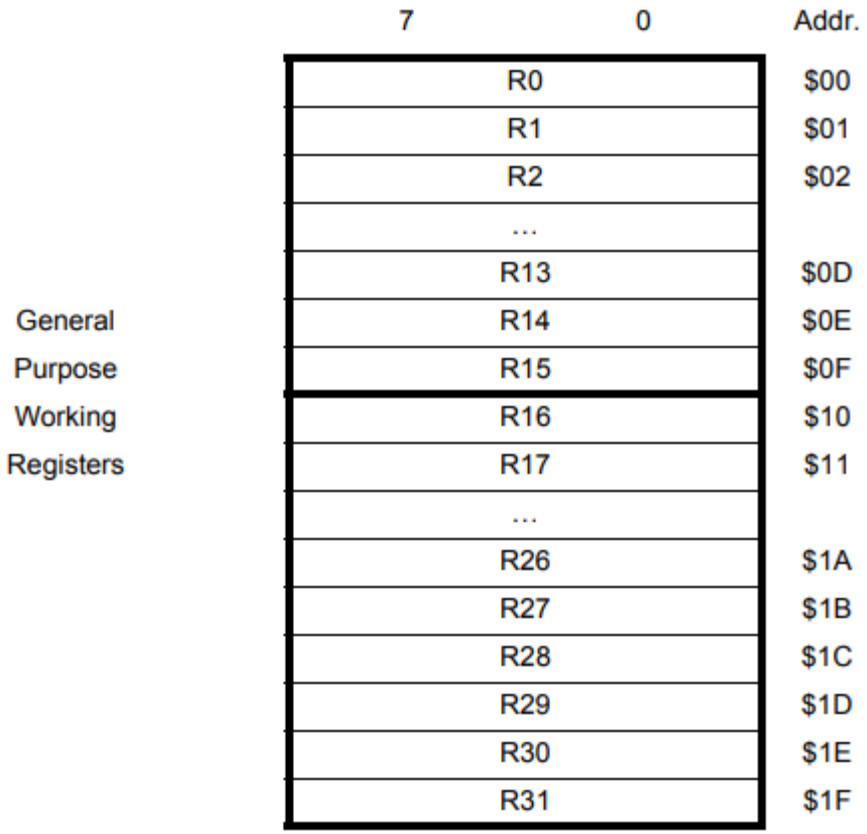
AVR Registers

- AVR Registers:
  - Two Types:
    - General Purpose:
    - Special Purpose:
      - Three Registers:
        - » Program counter
        - » Stack Pointer
        - » Status Register

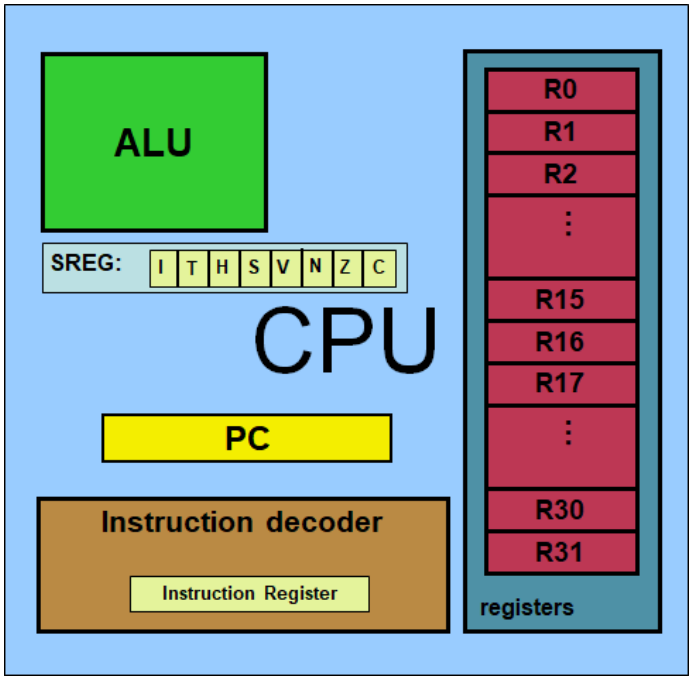


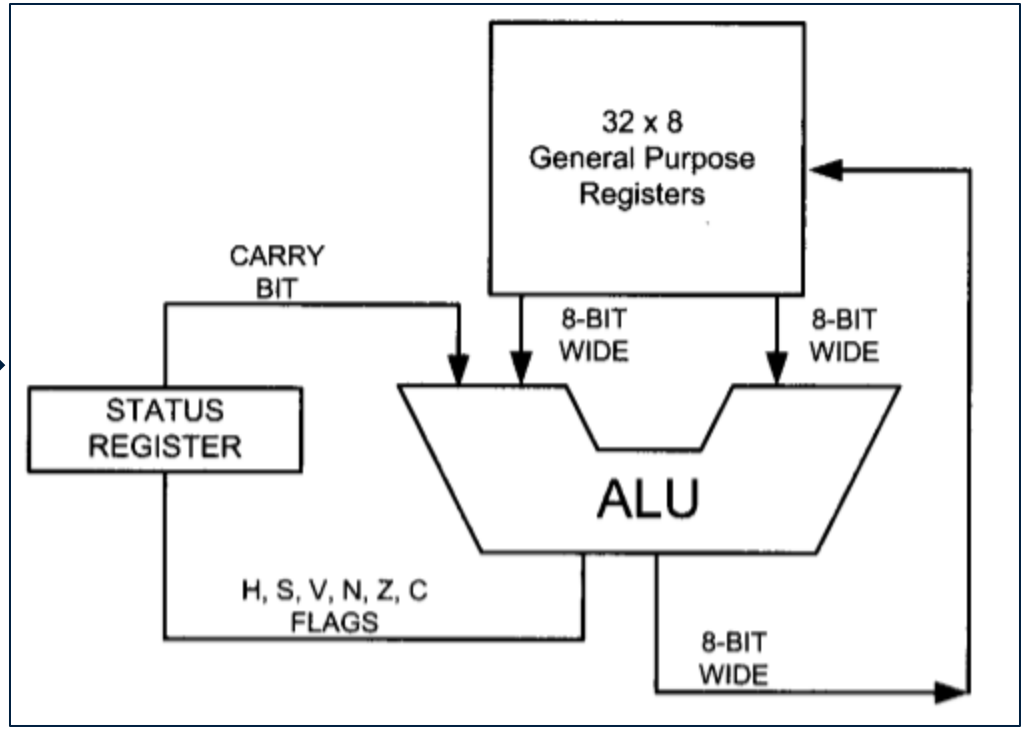
AVR Registers: General Purpose

- 32 general purpose registers having storage capacity of 8-Bits
- Named as R0,R1,R2 to R31.
- Register 0 to 15 & 16 to 31 are different.
- Can store both Data & Addresses.



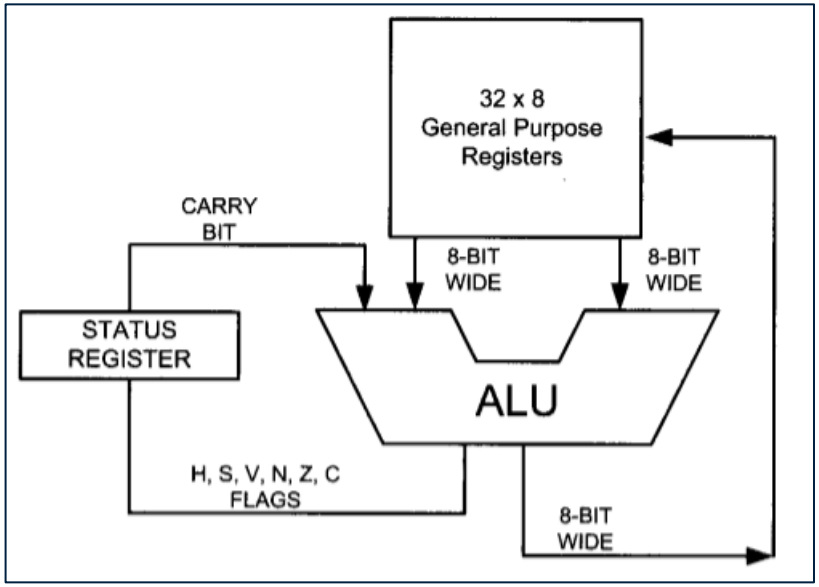
General Purpose Registers and ALU





General Purpose Registers and ALU

- The fast-access Register File contains:
  - 32 x 8-bit general purpose working registers
  - with a single clock cycle access time.
    - This allows single-cycle Arithmetic Logic Unit (ALU) operation.
    - In a typical ALU operation:
      - two operands are output from the Register File
      - the operation is executed,
      - and the result is stored back in the Register File
        - » All in one clock cycle.



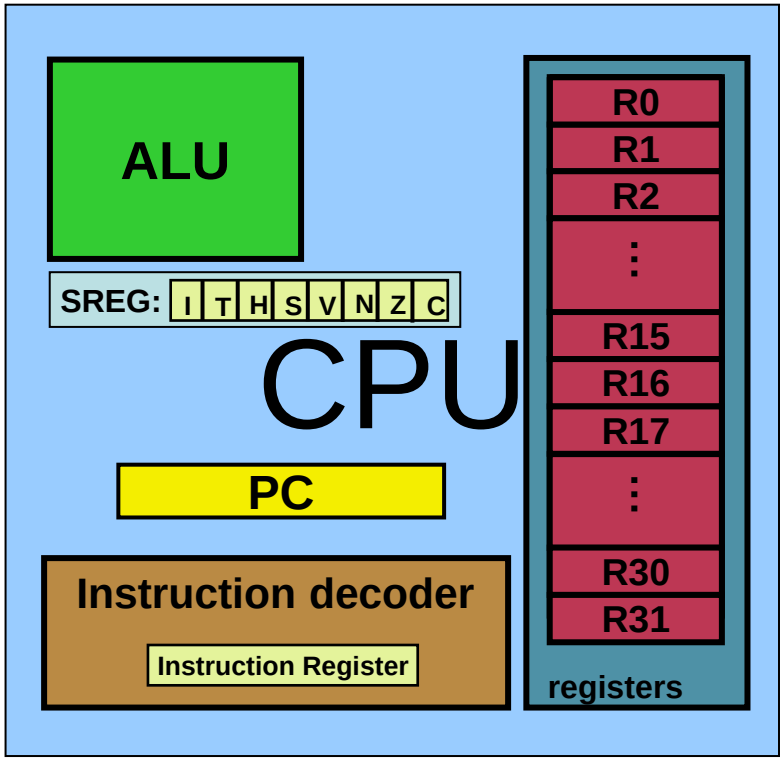
Some Simple Instructions

Loading Values into the General Purpose Registers

LDI (**Load Immediate**)

- LDI Rd, k
  - Its equivalent in high level languages:

    Rd = k (Between 16-31)
- Example:
  - LDI R16,53
    - R16 = 53
  - LDI R19,$27
  - LDI R23,0x27
    - R23 = 0x27
  - LDI R23,0b11101100

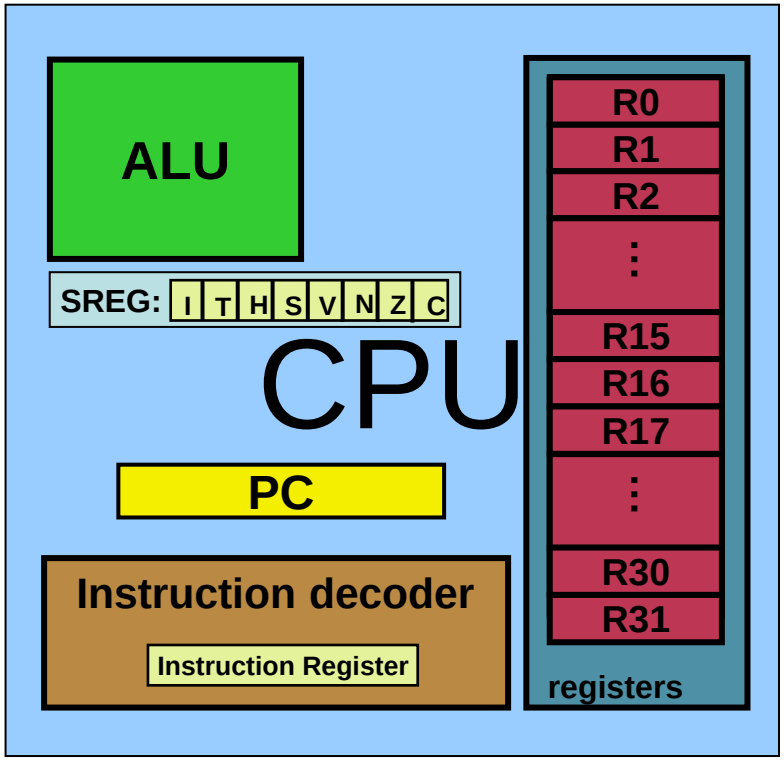


Arithmetic Operations

- There are some instructions for doing Arithmetic and logic operations; such as:

  ADD, SUB, MUL, AND, etc.

- ADD Rd,Rs
  - Rd = Rd + Rs
  - Example:
  - ADD R25, R9
    - R25 = R25 + R9
  - ADD R17,R30
    - R17 = R17 + R30



First Simple Program

- Write a program that calculates 19 + 95

```
LDI R16, 19     ;R16 = 19
LDI R20, 95     ;R20 = 95
ADD R16, R20    ;R16 = R16 + R20
```

First Simple Program -> Extended

- Write a program that calculates 19 + 95 + 5

```
LDI     R16, 19         ;R16 = 19
LDI     R20, 95         ;R20 = 95
LDI     R21, 5          ;R21 = 5
ADD     R16, R20        ;R16 = R16 + R20
ADD     R16, R21        ;R16 = R16 + R21
```

```
LDI     R16, 19         ;R16 = 19
LDI     R20, 95         ;R20 = 95
ADD     R16, R20        ;R16 = R16 + R20
LDI     R20, 5          ;R20 = 5
ADD     R16, R20        ;R16 = R16 + R20
```

General Purpose Register Rules

- Moving Larger Value will cause an error
- Values moved between 0 – F will cause the rest of the bits to be ZERO
- Hex numbers requires **0x** or **$** in front
- For Comments **;**

Reading Assignment

- The AVR Microcontroller and Embedded Systems: Using Assembly and C by Mazidi et al., Prentice Hall
  - Chapter 2: 2.1

THANK YOU