Assembly Language Examples for ATmega328P

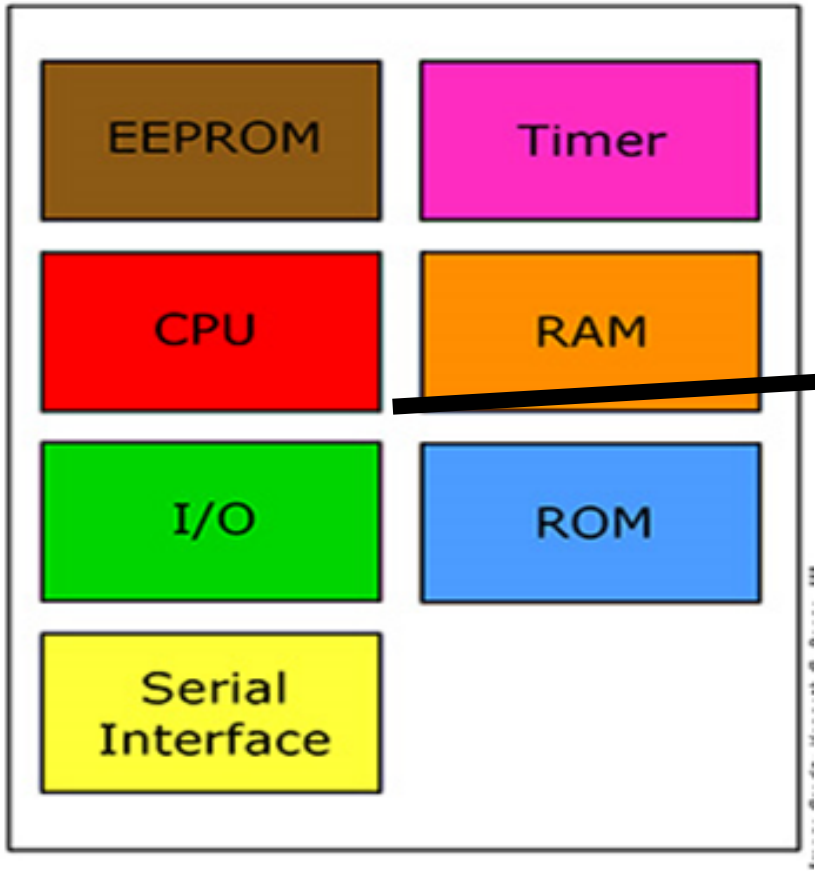
Why Assembly Language?

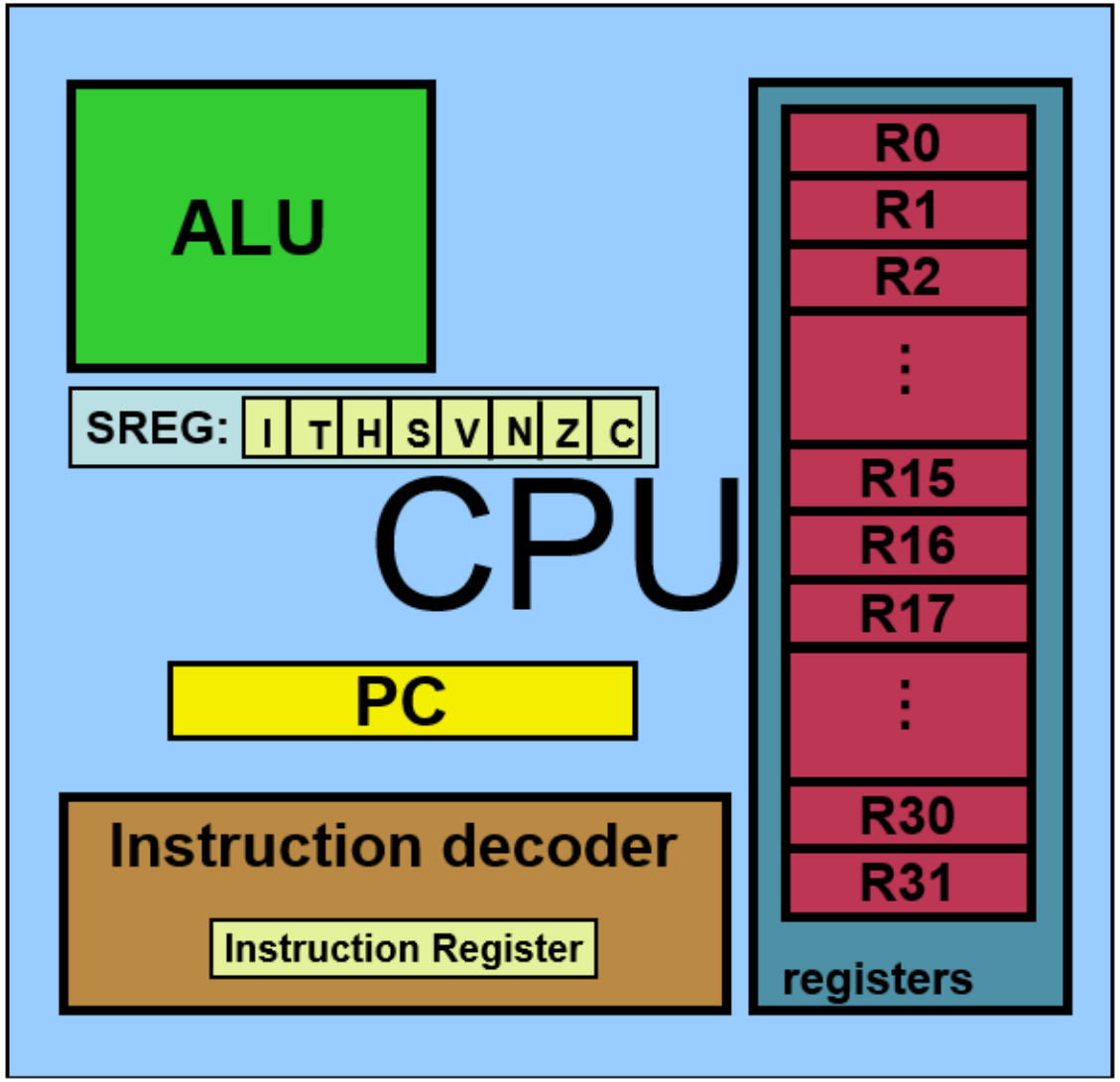
Commonly Assembly language is used -

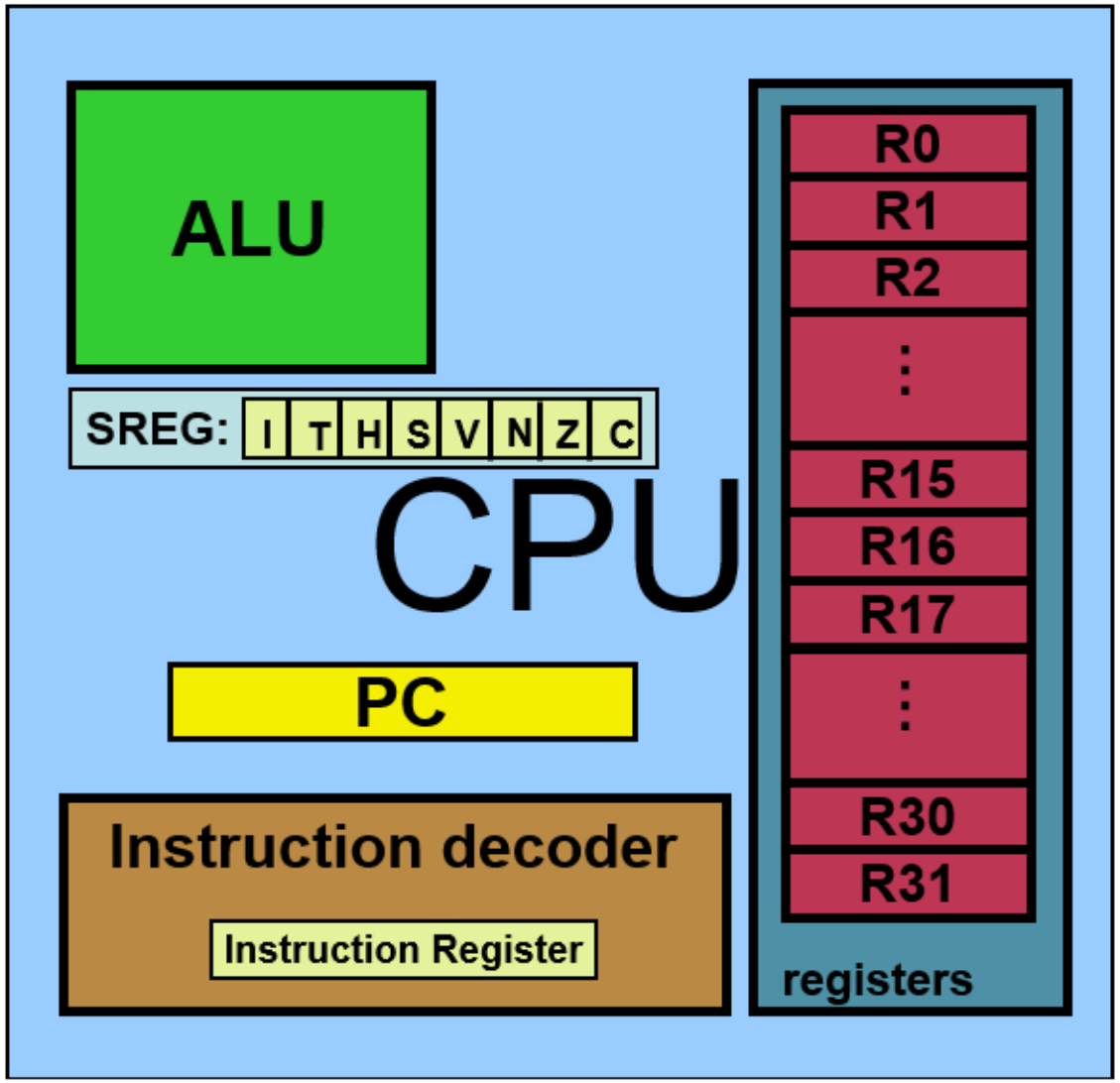
- ❑ to directly manipulate hardware
- ❑ to access specialized processor instructions
- ❑ to access machine-dependent registers and I/O
- ❑ Programs written in assembly can execute faster
- ❑ Assembly access is also sometimes used when creating bootloaders
- ❑ Assembly gives a more in-depth view of computer systems.
- ❑ to evaluate critical performance issues

ATEMGA328P CPU

Microcontroller: CPU on a single chip.







AVR Architecture of CPU

- ALU
- 32 General Purpose Register (each 8-bit)
- Program Counter (PC)
- Instruction Decoder
- Status Register (SREG)

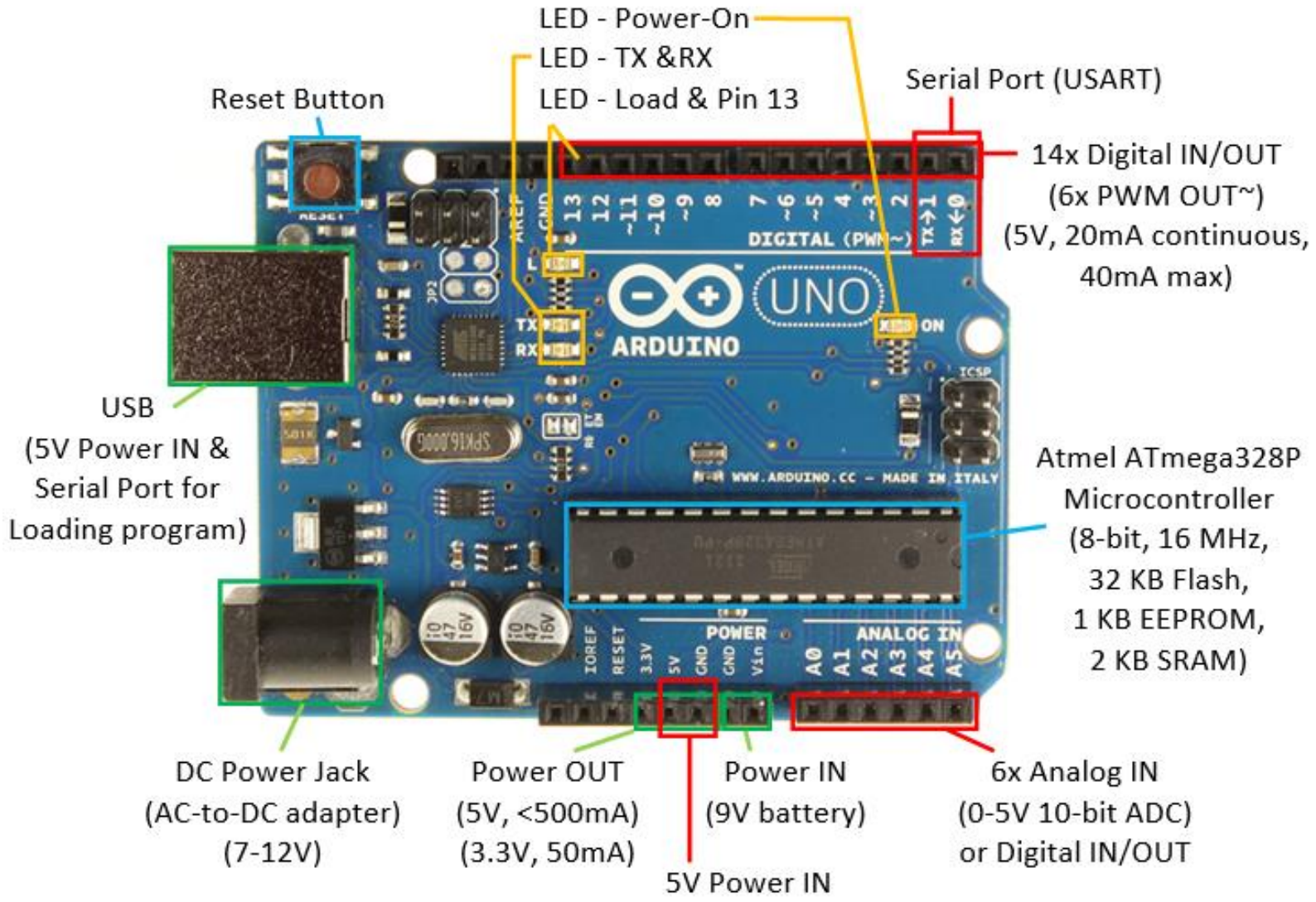
AVR is a family of microcontrollers developed since 1996 by Atmel, acquired by Microchip Technology in 2016. Atmel says that the name AVR is not an acronym and does not stand for anything in particular. However, it is commonly accepted that AVR stands for -

AVR = Alf and **V**egard's **R**ISC processor

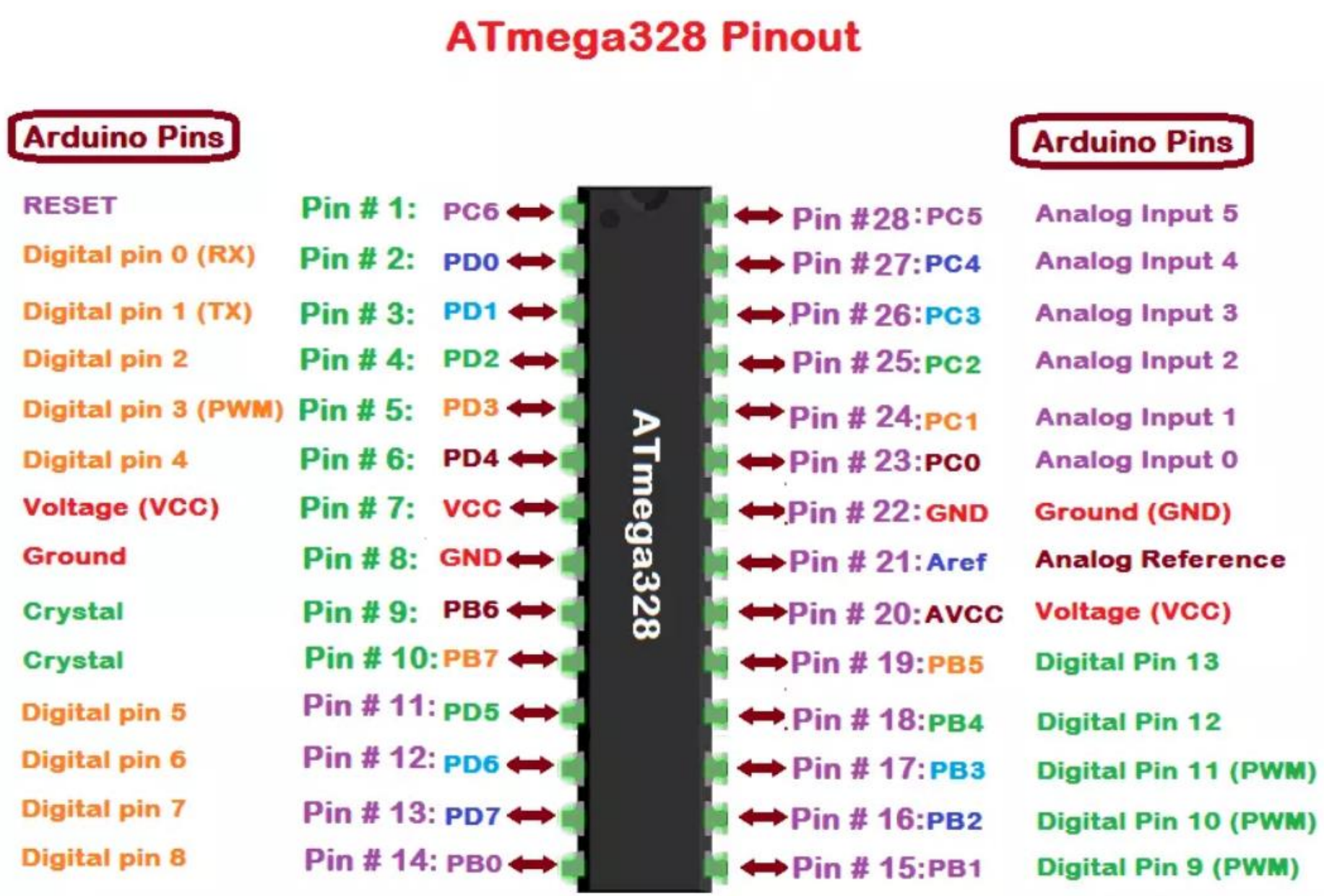
Status Register

- C = Carry Flag, This flag is set whenever there is a carry out from the D7 bit after an arithmetic operation(Addition, subtraction, increment, decrement etc). This flag bit is affected after an 8-bit addition or substruction.
- Z = Zero Flag, This flag is affected after an arithmetic or logic operation. If the result is zero than Z = 1, else Z = 0.
- N = Negative Flag, It reflects the result of an arithmetic operation. If the D7 bit of the result is zero then N = 0 and the result is positive. Else N = 1 and the result is negative.
- V = Overflow Flag,
- S = Sign Flag,
- H = Half Carry Flag, This bit is set if there is a carry from D3 to D4 bit after ADD or SUB instruction.
- T = Bit Copy Storage. Used as a temporary storage for bit. It can be used to copy a bit from a GPR to another GPR.
- I = Global Interrupt Enable

Overview of Arduino UNO Board



ATmega328P Pinout



Data Direction Register Method

To assign **input- 0**

To assign **output- 1**

DDRB= 0x 05;

Or DDRB= 0b 00000101; PB0 and PB2 are output pin while rest are input.

To write High-1

To write Low-0

PORTB= 0x01;

PORTB= 0b00000001; write PB0 as high.

DDRB Register

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	0	0	0	1	0	1

PORTB Register

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	0	0	0	0	0	1

Instruction set summary

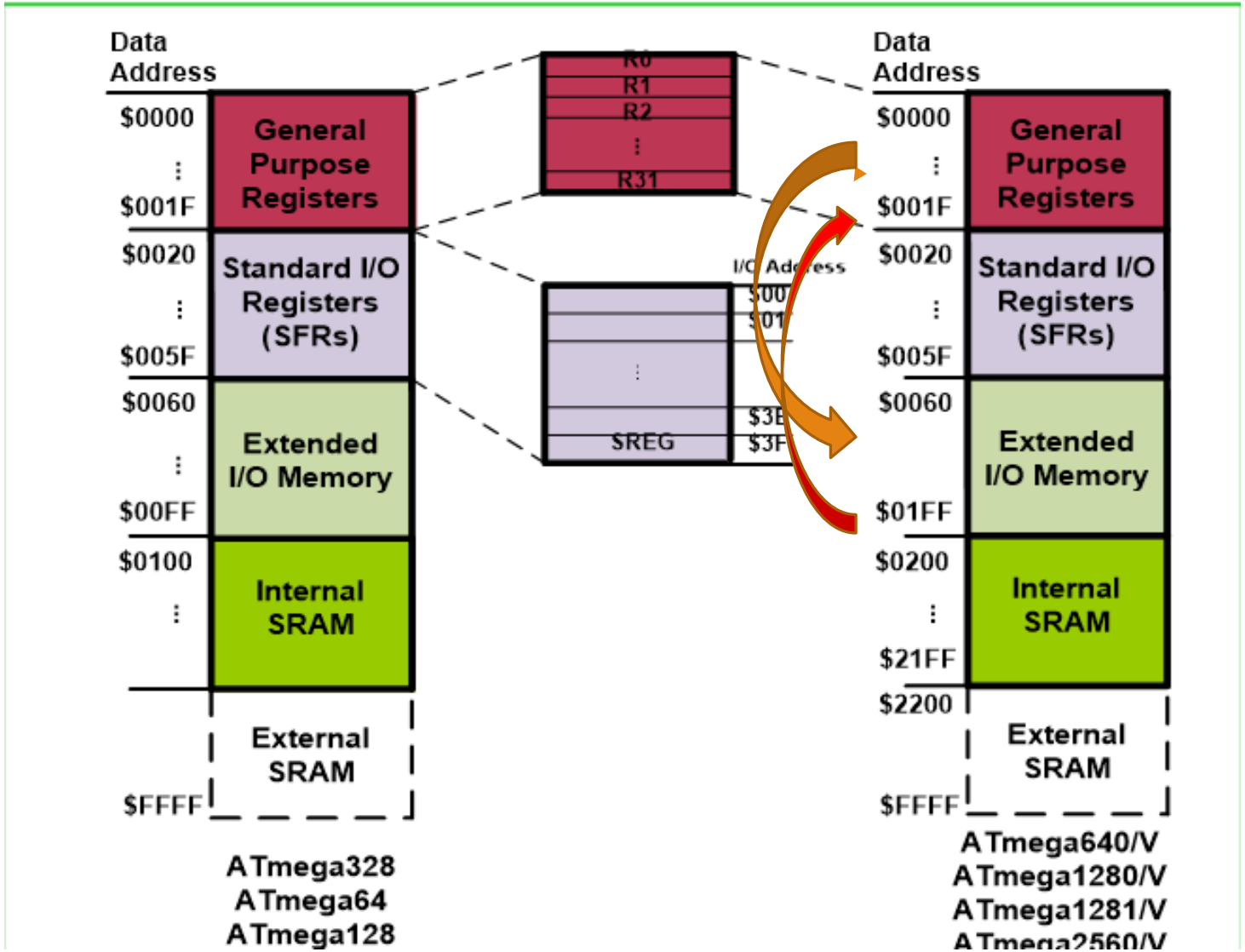
Mnemonics	Operands	Description	Operations	Flags	Clocks
ADD	Rd,Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd,Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
SUB	Rd,Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd,k	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
AND	Rd,Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
OR	Rd,Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 - Rd	Z,C,N,V,H	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
MUL	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2

Instruction set summary

Mnemonics	Operands	Description	Operations	Flags	Clocks
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LDS	Rd, k	Load Direct from data space location k	Rd← [k]	None	1
IN	Rd, P	From In Port P/ address to Rd register	Rd ← P	None	1
OUT	P, Rr	From Rr register to Out Port P/ address	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
NOP		No Operation	None	1	
BREAK		Break	For On-chip Debug Only	None	N/A

Instruction set summary

Mnemonics	Operands	Description	Operations	Flags	Clocks
STS	k, Rr	Store Rr to data space location k.	k ← Rr	None	1
SBI	A, b	Sets a specified bit in an I/O Register	Sets the b no. bits of A register	None	1
CBI	A, b	Resets a specified bit in an I/O Register	Resets the b no. bits of A register	None	1
SBIC	A, b	Skip if Bit in I/O Register is Cleared/ Reset	If b no bit of A register is 0, then skip PC by 2 or 3.	None	1
SBIS	A, b	Skip if Bit in I/O Register is set	If b no bit of A register is 1, then skip PC by 2 or 3.	None	1
RJUMP	K	Relative jump	PC← PC+K+1	None	1



STS 0x60, R0

LDS R31, 0x01FF

SBI DDRB, 3;

DDRB Register

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	0	0	0	1	0	0

CBI DDRB, 5;

DDRB Register

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
1	1	1	0	1	1	1	1

SBIC R7, 5;

R7 Register

8	7	6	5	4	3	2	1
1	1	1	0	1	1	1	1

PC=PC+2 or 3

SBIS R8, 5;

R8 Register

PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
0	0	0	1	0	0	0	0

PC=PC+2 or 3

Mnemonics	Operands	Description	Operations
STS	k, Rr	Store Rr to data space location k.	k ← Rr
SBI	A, b	Sets a specified bit in an I/O Register	Sets the b no. bits of A register
CBI	A, b	Resets a specified bit in an I/O Register	Resets the b no. bits of A register
SBIC	A, b	Skip if Bit in I/O Register is Cleared/ Reset	If b no bit of A register is 0, then skip PC by 2 or 3.
SBIS	A, b	Skip if Bit in I/O Register is set	If b no bit of A register is 1, then skip PC by 2 or 3.
RJUMP	K	Relative jump	PC← PC+K+1

N.B: A program counter (PC) is **a register in a computer processor that contains the address (location) of the instruction being executed at the current time**

Example: State the contents of R20, R21, and data memory location of 0x120 after the following program.

```
LDI R20, 5;
LDI R21, 2;
ADD R20,R21;
ADD R20,R21;
STS 0X120, R20;
```

Solution:

```
LDI R20, 5;          //R20=5
LDI R21, 2;          R21=2
ADD R20,R21;         R20=R20+R21=5+2=7
ADD R20,R21;         R20= 7+2=9
STS 0X120, R20;      0x120 = 9
```

R20= 9

R21= 2

0x120= 9

Example: State the contents of RAM location of $212, $213, $214, $215 and $216 after the following program.

```
LDI R16, 0x99
STS 0x212, R16
LDI R16, 0x85
STS 0x213, R16
LDI R16, 0x3F
STS 0x214, R16
LDI R16, 0x63
STS 0x215, R16
LDI R16, 0x12
STS 0x216, R16
```

Solution	
0x212	0x99
0x213	0x85
0x214	0x3F
0x215	0x63
0x216	0x12

Thank You