به نام خدا

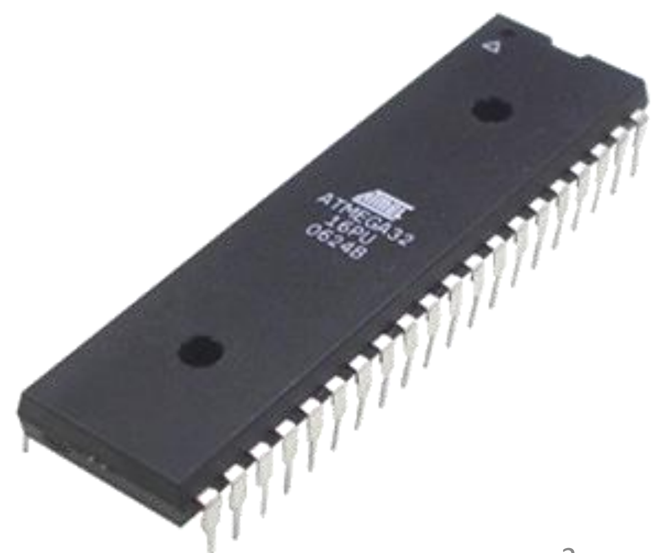
آشنایی با معماری AVR

زبان اسمبلی AVR

Dr. Aref Karimiafshar
A.karimiafshar@iut.ac.ir

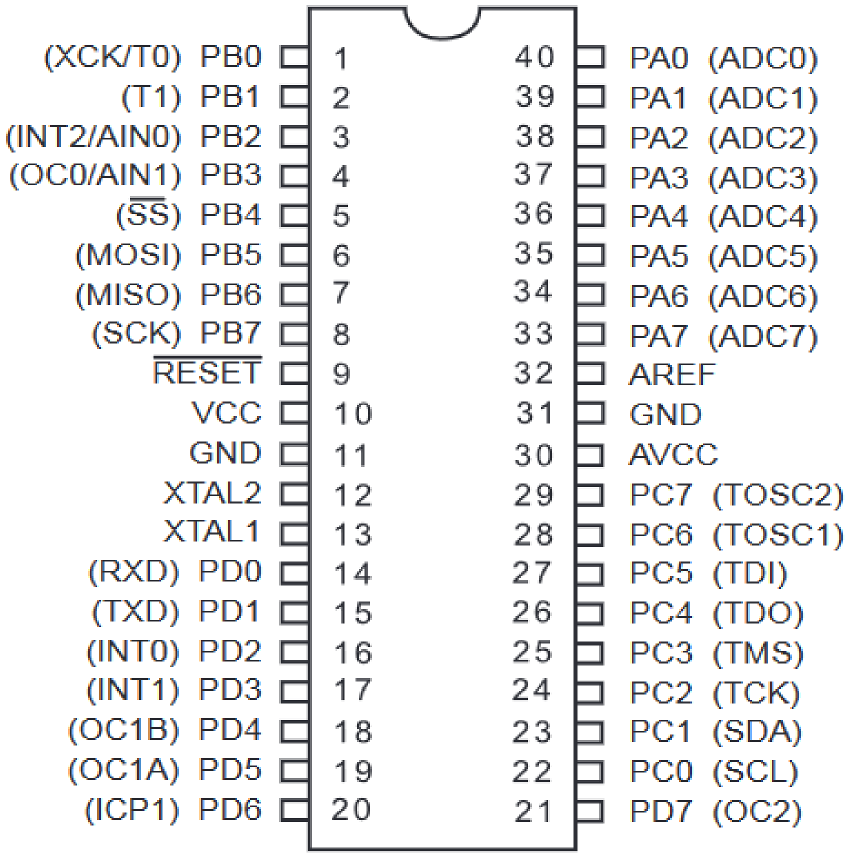
ATmega32 Features

- High-performance, Low-power 8-bit μC
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash program memory
  - 1024Bytes EEPROM
  - 2Kbytes Internal SRAM
- Operating Voltages
  - 2.7V - 5.5V for ATmega32L
  - 4.5V - 5.5V for ATmega32
- Speed Grades
  - 0 - 8MHz for ATmega32L
  - 0 - 16MHz for ATmega32

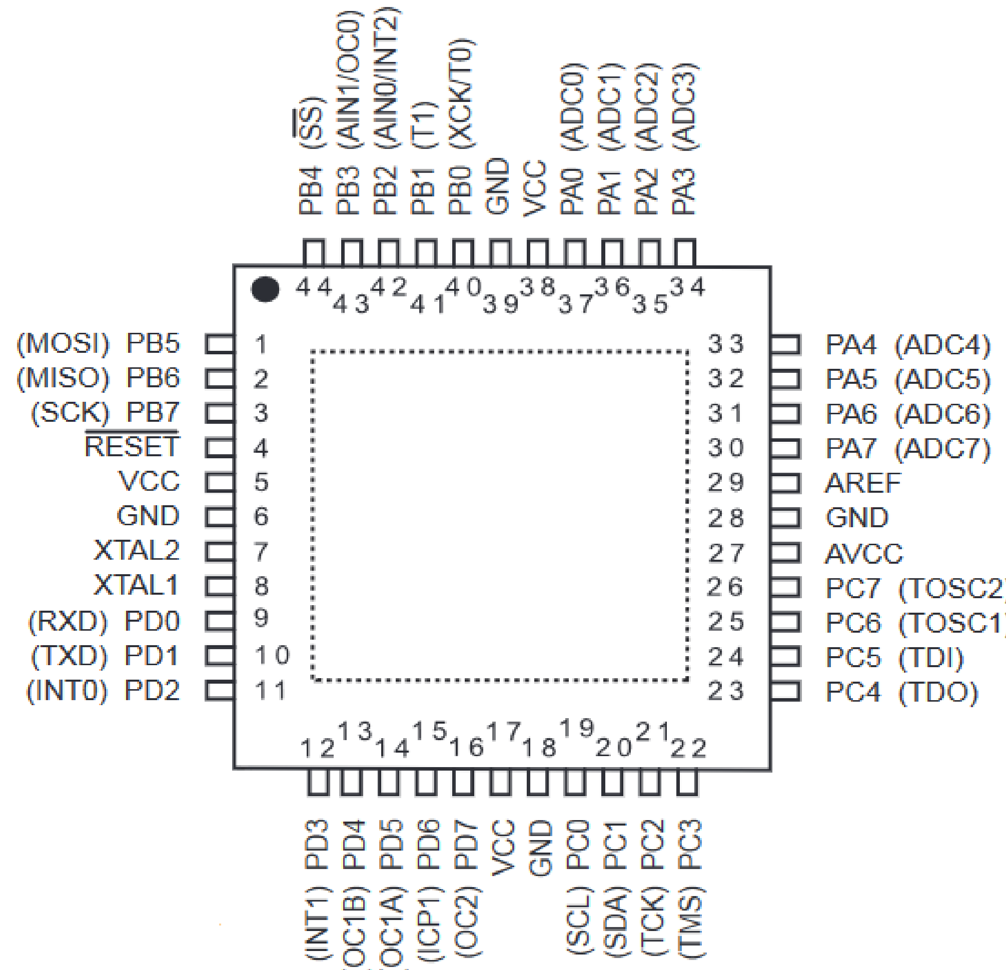


Pinout ATmega32

PDIP



TQFP/MLF

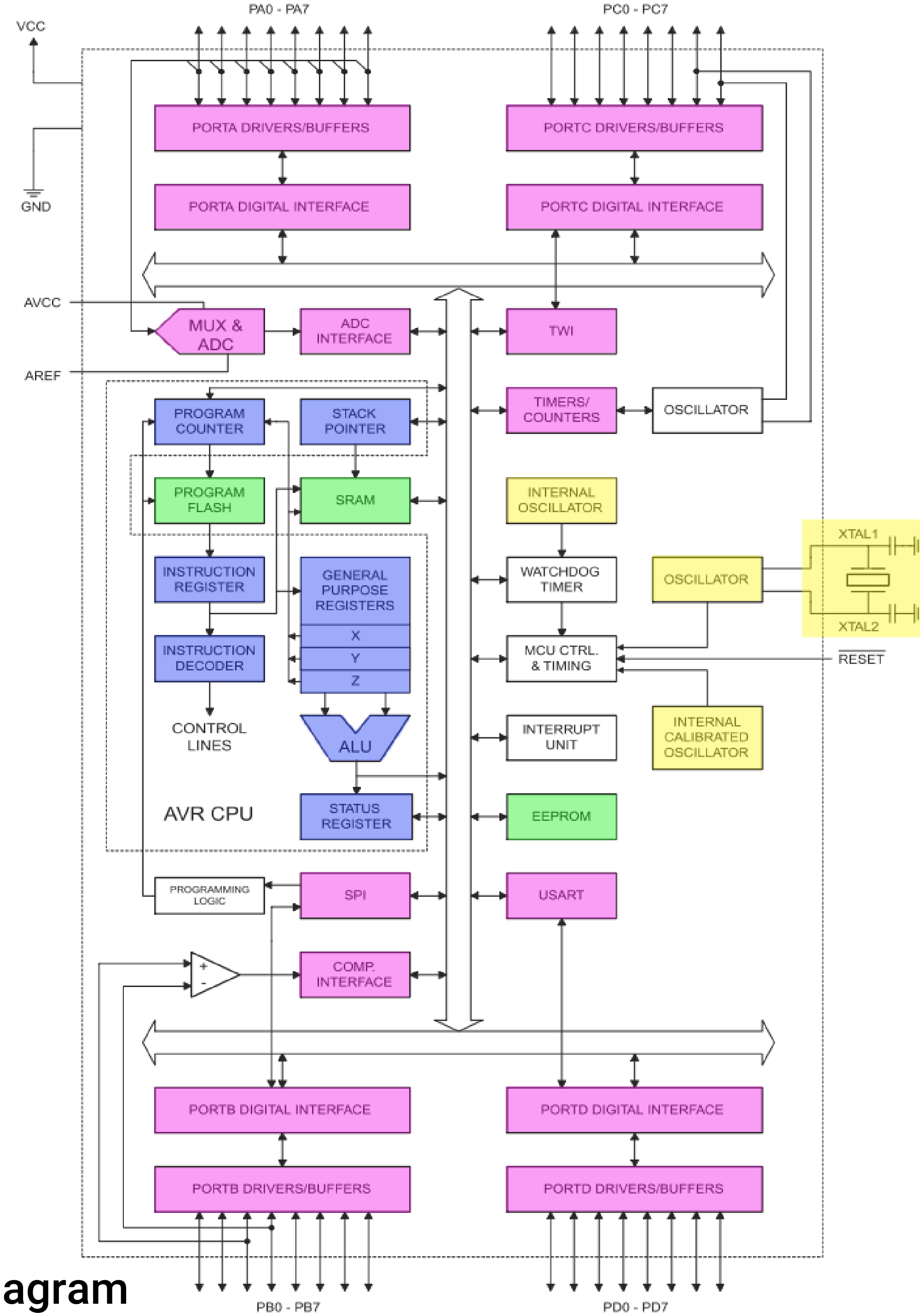


Pin Descriptions

Pin	Descriptions
VCC	Digital supply voltage
GND	Ground
Port A (PA7..PA0)	1. Port A serves as the analog inputs to the A/D Converter. 2. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used.
Port B (PB7..PB0)	1. Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). 2. Port B also serves the functions of various special features of the ATmega32.
Port C (PC7..PC0)	1. Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). 2. Port C also serves the functions of the JTAG interface and other special features of the ATmega32.

Pin Descriptions

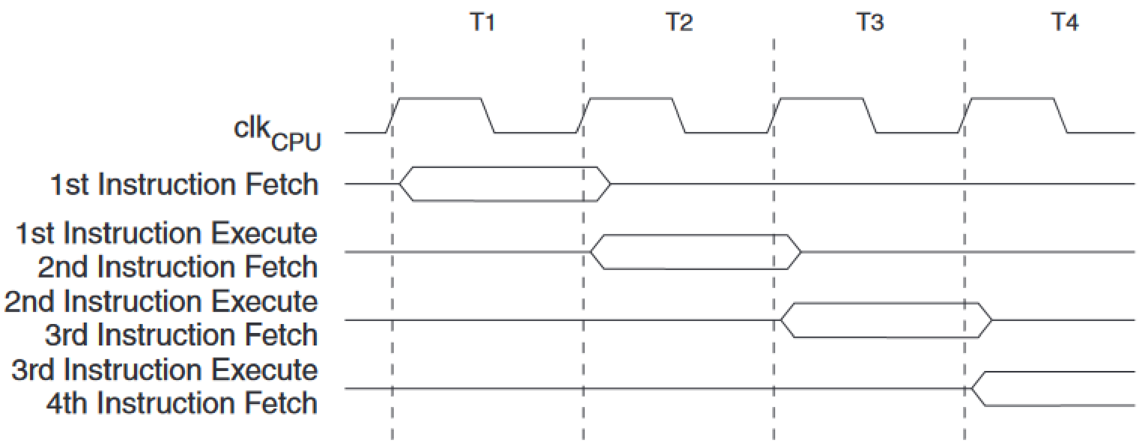
Pin	Descriptions
Port D (PD7..PD0)	1. Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). 2. Port D also serves the functions of various special features of the ATmega32.
$\overline{RESET}$	Reset Input.
XTAL1	Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.
XTAL2	Output from the inverting Oscillator amplifier.
AVCC	AVCC is the supply voltage pin for Port A and the A/D Converter.
AREF	AREF is the analog reference pin for the A/D Converter.



ATmega32 Block Diagram

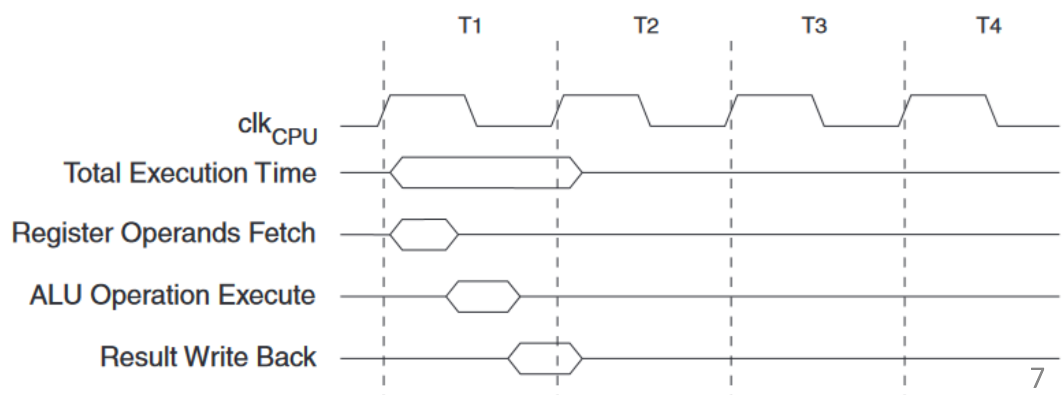
ATmega32 Features

- To maximize performance and parallelism
  - AVR uses a Harvard architecture (**separate memories and buses for program and data**)
- Instructions in the program memory are executed with a single level pipelining



While one instruction is being executed, next instruction is pre-fetched from the program memory

This concept enables instructions to be executed in every clock cycle



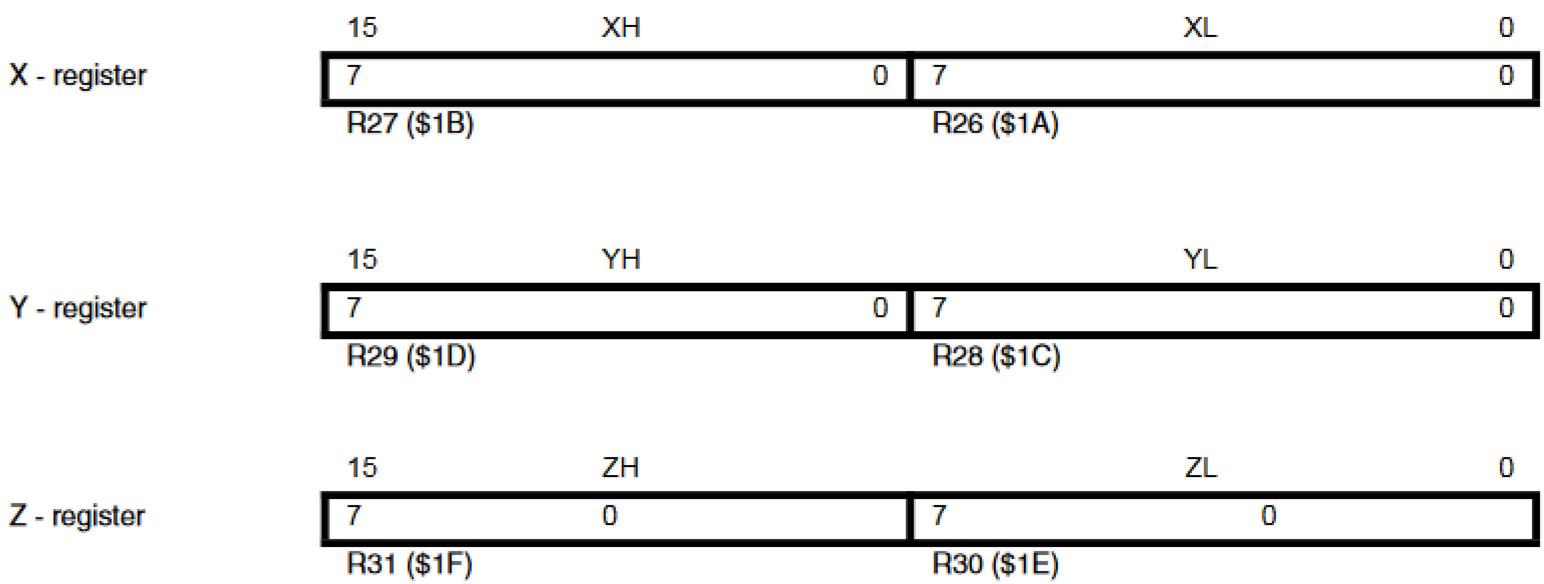
Register File

- Fast-access Register File
  - Contains 32 × 8-bit General Purpose working Registers (GPR)
  - ALU operates in direct connection with all the 32 GPR
  - with a single clock cycle access time
- Each register is assigned a data memory address
  - Mapping them directly into the first 32 locations of the user Data Space
  - Although not being physically implemented as SRAM locations

7 … 0	Addr.	
R0	$00	
R1	$01	
R2	$02	
...		
R13	$0D	
R14	$0E	
R15	$0F	
R16	$10	
R17	$11	
...		
R26	$1A	X-register Low Byte
R27	$1B	X-register High Byte
R28	$1C	Y-register Low Byte
R29	$1D	Y-register High Byte
R30	$1E	Z-register Low Byte
R31	$1F	Z-register High Byte

X, Y, and Z-register

- Registers R26..R31 have some added functions to their general purpose usage
  - These registers are 16-bit address pointers for indirect addressing of the Data Space



Stack Pointer

- The Stack is mainly used for storing temporary data for
  – Storing local variables
  – For storing return addresses after interrupts and subroutine calls
- The Stack Pointer Register always points to the top of the Stack.
  – Note that the Stack is implemented as growing from higher memory locations to lower memory locations
  – This implies that a Stack PUSH command decreases the Stack Pointer
- AVR Stack Pointer is implemented as two 8-bit registers in I/O space

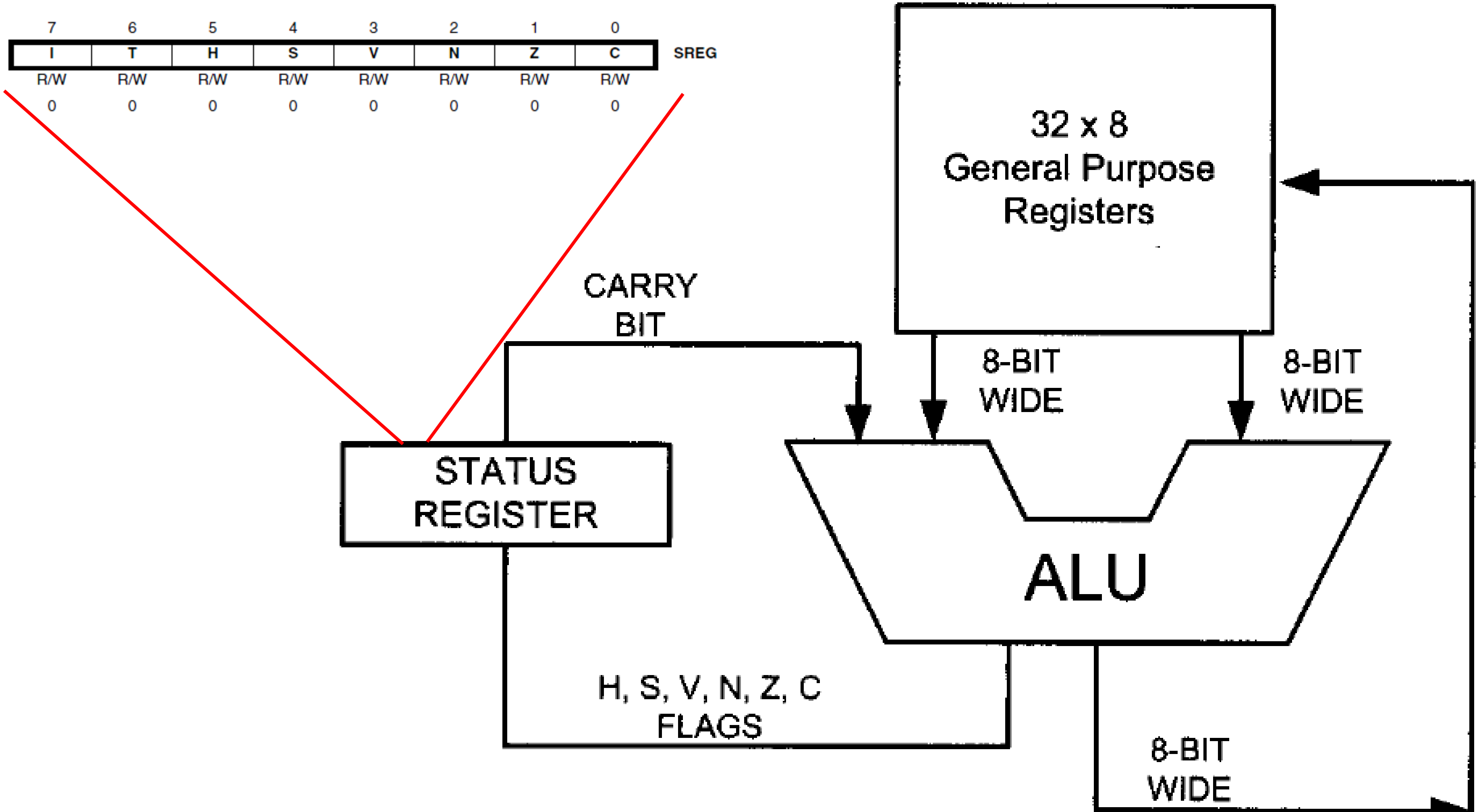
Bit	15	14	13	12	11	10	9	8	
	SP15	**SP14**	**SP13**	**SP12**	**SP11**	**SP10**	**SP9**	**SP8**	**SPH**
	SP7	**SP6**	**SP5**	**SP4**	**SP3**	**SP2**	**SP1**	**SP0**	**SPL**
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

ALU

- ALU operations
  - Between registers
  - Between a constant and a register
  - Single register operations can also be executed in the ALU
- After an arithmetic operation, the Status Register is updated
  - To reflect information about the result of the operation
    - Most recently executed arithmetic instruction

Bit	7	6	5	4	3	2	1	0	
	I	**T**	**H**	**S**	**V**	**N**	**Z**	**C**	**SREG**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ALU


7
6
5
4
3
2
1
0
I
T
H
S
V
N
Z
C
SREG
R/W
R/W
R/W
R/W
R/W
R/W
R/W
R/W
0
0
0
0
0
0
0
0
32 x 8
General Purpose
Registers
CARRY
BIT
8-BIT
WIDE
8-BIT
WIDE
STATUS
REGISTER
ALU
H, S, V, N, Z, C
FLAGS
8-BIT
WIDE

AVR Status Register (SREG)

- Bit 7 – I: Global Interrupt Enable
  - The Global Interrupt Enable bit must be set for the interrupts to be enabled.
- Bit 6 – T: Bit Copy Storage
  - Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit
- Bit 5 – H: Half Carry Flag
  - The Half Carry Flag H indicates a half carry in some arithmetic operations
- Bit 4 – S: Sign Bit
  - S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V
- Bit 3 – V: Two's Complement Overflow Flag
  - The Two's Complement Overflow Flag V supports two's complement arithmetic
- Bit 2 – N: Negative Flag
  - The Negative Flag N indicates a negative result in an arithmetic or logic operation
- Bit 1 – Z: Zero Flag
  - The Zero Flag Z indicates a zero result in an arithmetic or logic operation
- Bit 0 – C: Carry Flag
  - The Carry Flag C indicates a carry in an arithmetic or logic operation

Bit	7	6	5	4	3	2	1	0	
	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

ATmega32 Features

- Program flow is provided by
  - Conditional jump instruction
  - Unconditional jump instruction
  - Call instruction
- During interrupts and subroutine calls
  - The return address Program Counter (PC) is stored on the Stack
  - The Stack is effectively allocated in the general data SRAM
    - consequently the Stack size is only limited by the total SRAM size and the usage of the SRAM
- The data SRAM can easily be accessed
  - Through the five different addressing modes supported in the AVR architecture
- Most AVR instructions have a single 16-bit word format
- Every program memory address contains a 16- or 32-bit instruction

LDI – Load Immediate

- Loads an 8-bit constant directly to register 16 to 31

(i) Rd ← K

	Syntax:	Operands:	Program Counter:
(i)	LDI Rd,K	$16 \le d \le 31$, $0 \le K \le 255$	PC ← PC + 1

16-bit Opcode:

1110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

MOV – Copy Register

This instruction makes a copy of one register into another. The source register Rr is left unchanged, while the destination register Rd is loaded with a copy of Rr.

Operation:

(i) $Rd \leftarrow Rr$

	Syntax:	Operands:	Program Counter:
(i)	MOV Rd,Rr	$0 \le d \le 31, 0 \le r \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

0010	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

ADD – Add without Carry

Adds two registers without the C Flag and places the result in the destination register Rd

Operation:

(i) (i) $Rd \leftarrow Rd + Rr$

	Syntax:	Operands:	Program Counter:
(i)	ADD Rd,Rr	$0 \le d \le 31$, $0 \le r \le 31$	$PC \leftarrow PC + 1$

16-bit Opcode:

0000	11rd	dddd	rrrr

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

Words	1 (2 bytes)
Cycles	1

ADD – Add without Carry

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

H $Rd3 \cdot Rr3 + Rr3 \cdot \overline{R3} + \overline{R3} \cdot Rd3$

Set if there was a carry from bit 3; cleared otherwise.

S $N \oplus V$, for signed tests.

V $Rd7 \cdot Rr7 \cdot \overline{R7} + \overline{Rd7} \cdot \overline{Rr7} \cdot R7$

Set if two's complement overflow resulted from the operation; cleared otherwise.

N $R7$

Set if MSB of the result is set; cleared otherwise.

Z $\overline{R7} \cdot \overline{R6} \cdot \overline{R5} \cdot \overline{R4} \cdot \overline{R3} \cdot \overline{R2} \cdot \overline{R1} \cdot \overline{R0}$

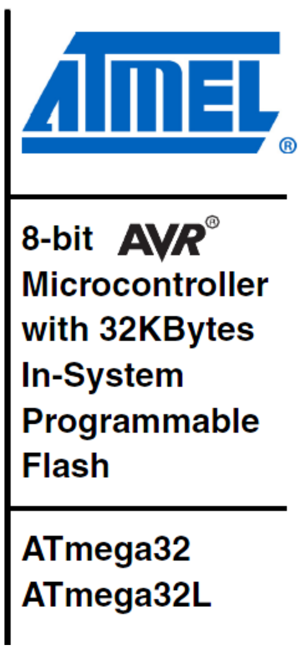
Set if the result is $00; cleared otherwise.

C $Rd7 \cdot Rr7 + Rr7 \cdot \overline{R7} + \overline{R7} \cdot Rd7$

Set if there was carry from the MSB of the result; cleared otherwise.

چرا اسمبلی؟!

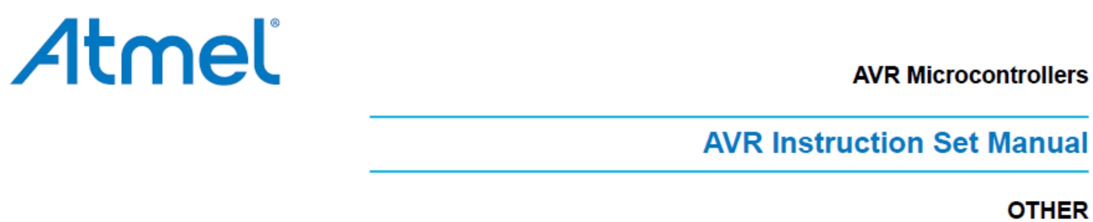
- کار با زبان اسمبلی درک عمیق‌تری از سخت‌افزار به شما می‌دهد.
- برنامه‌نویس سخت‌افزار اگر از جزئیات سخت‌افزار به درستی اطلاع نداشته و آن را خوب درک نکرده باشد، در کارهای حرفه‌ای با مشکل مواجه خواهد بود.
- ارجاع به منابع اصلی شرکت فراهم کننده
  - ATmega32 Datasheet
  - AVR Instruction Set Manual



ATMEL®

8-bit AVR® Microcontroller with 32KBytes In-System Programmable Flash

ATmega32
ATmega32L



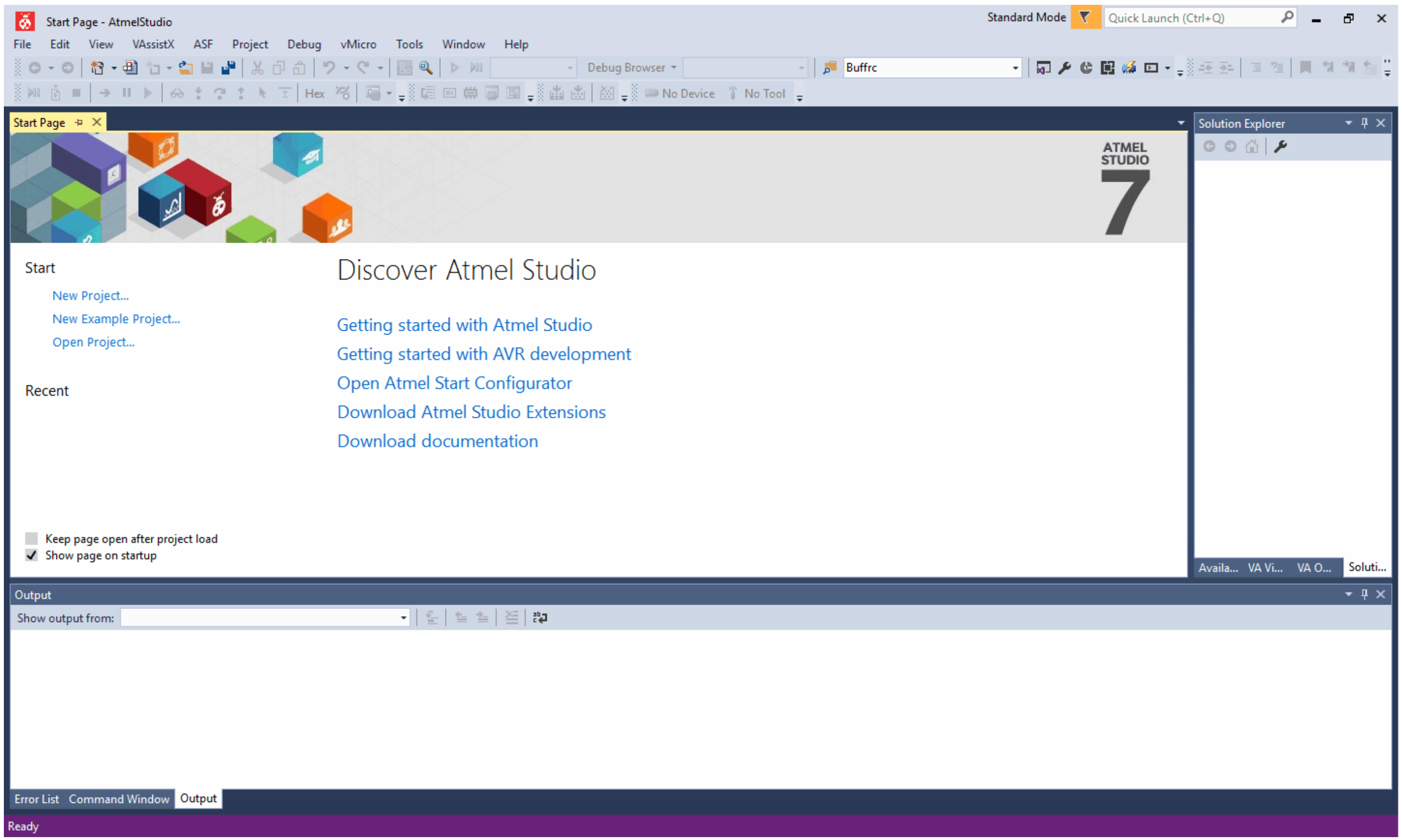
Atmel®

AVR Microcontrollers

AVR Instruction Set Manual

OTHER

Atmel Studio (Microchip Studio)



نصب نرم‌افزار و شروع به برنامه‌نویسی

پایان

موفق و پیروز باشید