به نام خدا

آشنایی با معماری AVR

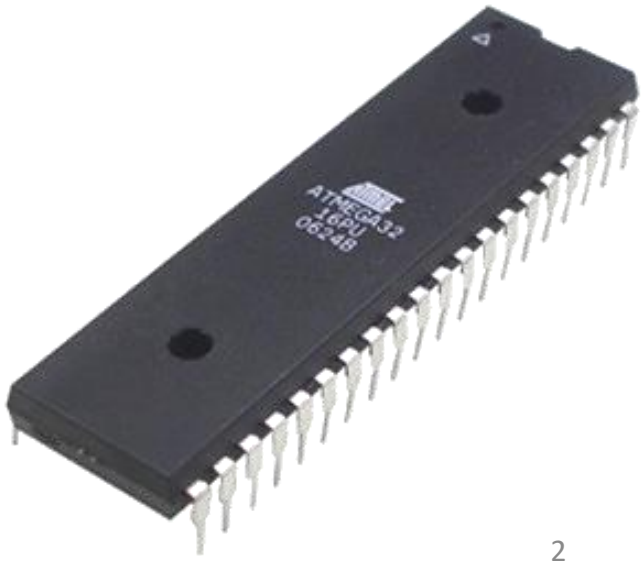
زبان اسمبلی AVR

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

--

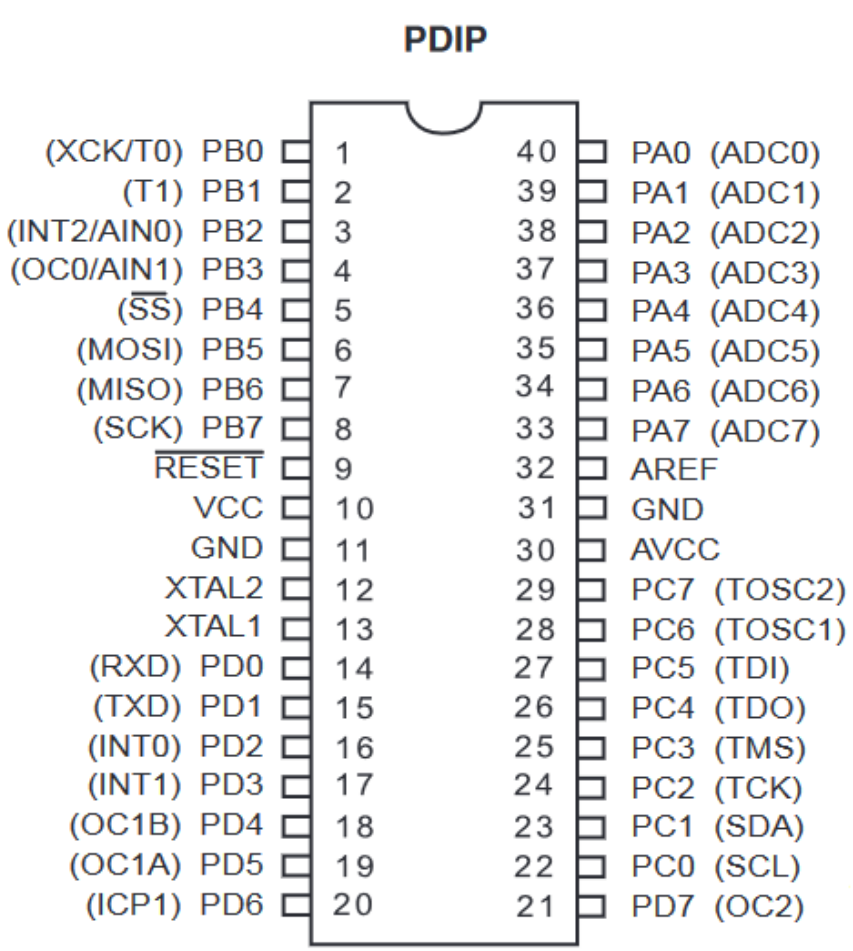
ATmega32 Features

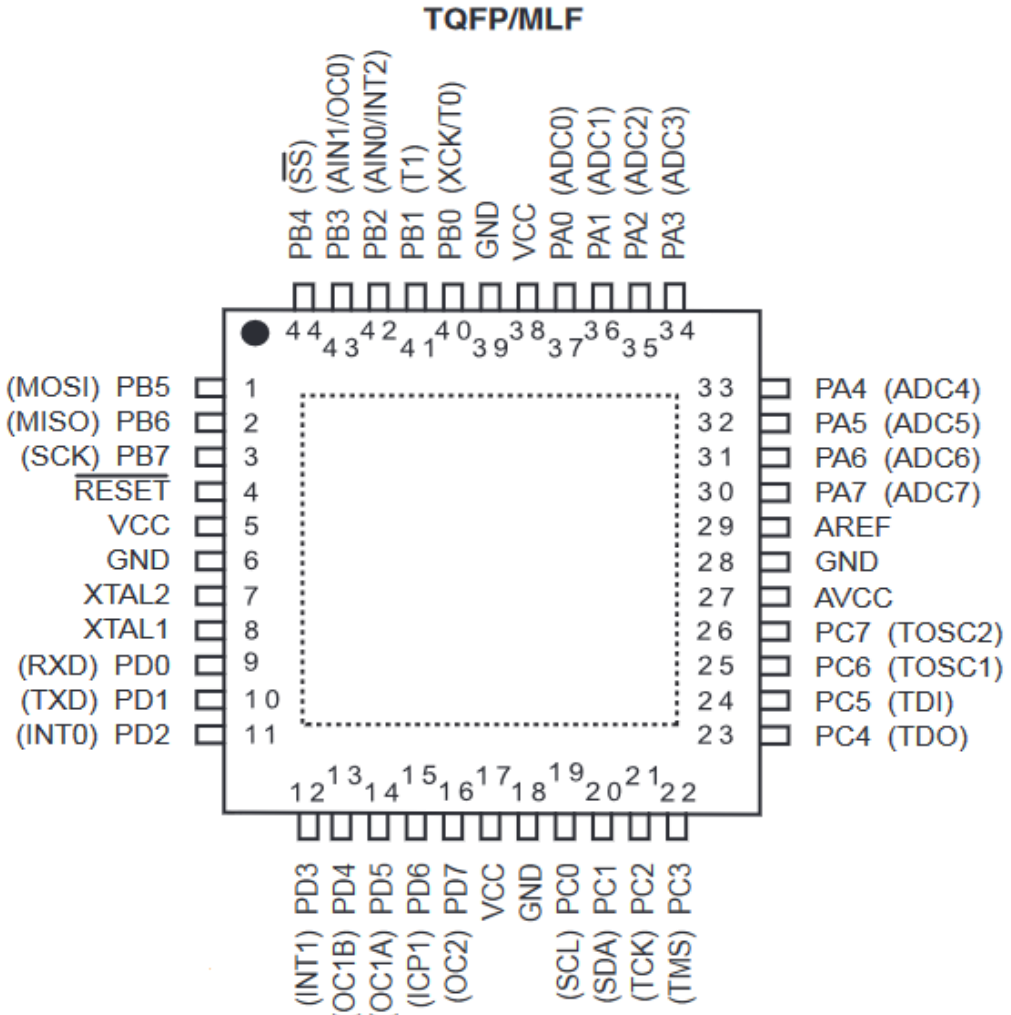
- High-performance, Low-power 8-bit μC
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 × 8 General Purpose Working Registers
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 32Kbytes of In-System Self-programmable Flash 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



-
این یک میکروکنترلر بسیار پرکاربرد با امکانات مناسب هستش و یک میکروکنترلر با کارایی بالا و توان مصرفی کمی داره و خود تراشه هم دو نوع داره ینی ATmega32 , ATmega32L
معماری اش RISC است
و 131 دستور داره و اکثر اینا در یک سیکل ساعت اجرا میشن
32 تا رجیستر همه منظوره 8 بیتی داره و یک ضرب کننده هم داخل خودش داره
حافظه برنامه اون 32 کیلوبایت است که 32 که داخل اسمش است اشاره به این حافظه برنامه داره
1024 بایت EEPROM داره و 2 کیلوبایت هم SRAM داره
ولتاژ کاری اون توی نسخه های low power بین 2.7 تا 5.5 است و توی سری های معمولی بین 4.5 تا 5.5 هستش
فرکانسی که توی low power داریم بین 0 تا 8 مگا هرتز است و برای سری معمولی بین 0 تا 16 مگا هرتز است

Pinout ATmega32





توی این اسلاید چینش پایه ها و کاربردهای پایه های مختلف رو نشون داده
ATmega32 در دو نوع PDIP , TQFP/MLF در بازار عرضه میشه
در نوع DIP اون 40 پایه داریم که هر سمت اون 20 پایه قرار داره و 4 پورت A , B , C , D داریم و یکسری پایه های کنترلی و تغذیه داریم
در نوع QTP/ MLF ما 44 پایه داریم و باز همون پورت A,B,C, D + پایه های تغذیه و کنترلی داریم
و توی شکل یک چیزی است که شروع پایه ها رو مشخص میکنه: مشخصه ببین

Pin Descriptions

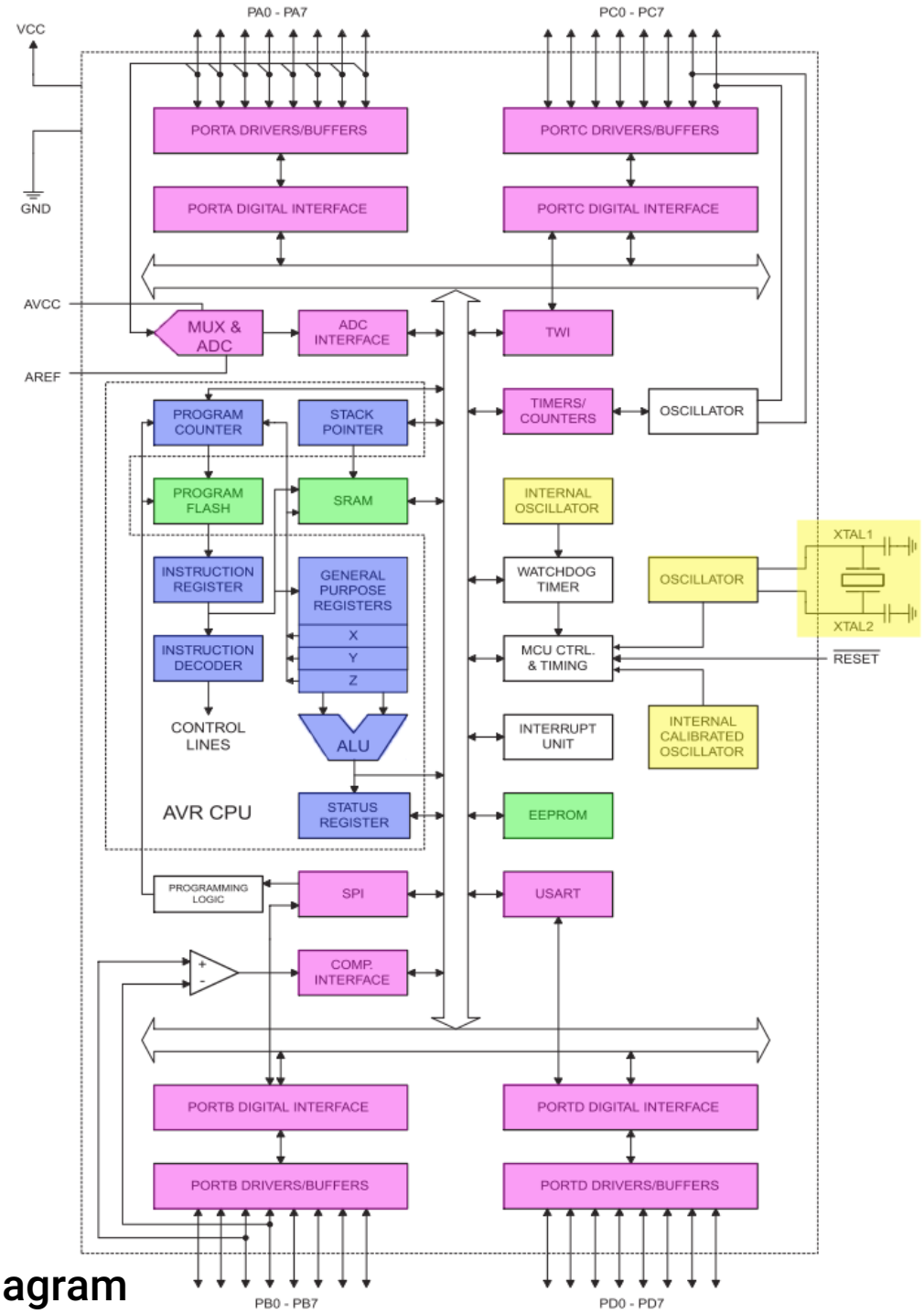
Pin	Descriptions
VCC	Digital supply voltage
GND	Ground
Port A (PA7..PA0)	1. Port A serves as the analog inputs to the A/D Converter. 2. Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used.
Port B (PB7..PB0)	1. Port B is an 8-bit bi-directional I/O port with internal 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.

پایه 10 مربوط به vccمیشه و VCC تغذیه اصلی تراشه رو فراهم می کنه و GND هم زمین این تراشه
پورت A که از 7 تا صفر است این به عنوان پورت A است و دوتا کاربر داره:
1- به عنوان یکسری ورودی های انالوگ برای مبدل انالوگ به دیجیتال اون حساب میشه پس ورودی انالوگ می دیم به این تراشه و خروجی دیجیتال می گیریم
2- پورت A به عنوان یک پورت دو جهته ورودی دیجیتال هم استفاده میشه ینی اگر پورت A تنظیم نکنیم که به عنوان یک مبدل دیجیتال عمل بکنه این به عنوان پورت دو جهته دیجیتال می تونه مورد استفاده قرار بگیره
پورت B :
1- به عنوان ورودی دو جهته دیجیتال است
2- پورت B همچنین عملکردهای مختلف ویژگی های خاص ATmega32 را انجام می دهد.
پورت C:
1- پورت C یک پورت ورودی/خروجی دو جهته 8 بیتی با مقاومت های کششی داخلی (انتخاب شده برای هر بیت) است.
2- پورت C همچنین عملکردهای رابط JTAG و سایر ویژگی های خاص ATmega32 را ارائه می دهد

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.

پورت D:
1- هم به عنوان یک پورت دیجیتال ورودی می تونه مورد استفاده قرار بگیره
2- یکسری عملکرد های خاص دیگری
پورت not RESET:
یک پایه است برای ریست و در پایه شماره 9 است و اگر این پایه low بشه کل تراشه و حافظه اون ریست میشه البته به جز حافظه های دائمی
پورت XTAL2 , XTAL1 که به عنوان ورودی و خروجی oscillator میشه ازشون استفاده کرد که پایه های 12 و 13 میشه - هر چند که میدونیم ATmega32 خودش یک oscillator داخلی داره
پورت AVCC:
AVCC پایه ولتاژ تغذیه پورت A و مبدل A/D است. - این ها برای مبدل انالوگ به دیجیتال استفاده میشه
پورت AREF:
ولتاژ رفرنسی است که برای این مبدل مورد استفاده قرار خواهد گرفت - AREF پین مرجع آنالوگ برای مبدل A/D است

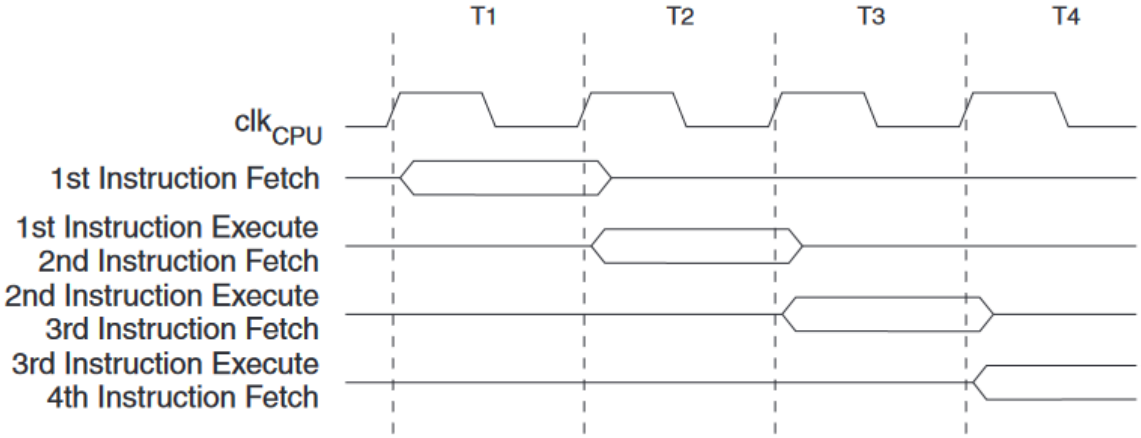


ATmega32 Block Diagram

معماری داخلی این تراشه:
پورت A,B,C,D از طریق اینترفس ها با بافرها و درایور ها متصل میشن به باس اصلی
شکل رو خودت ببین!!

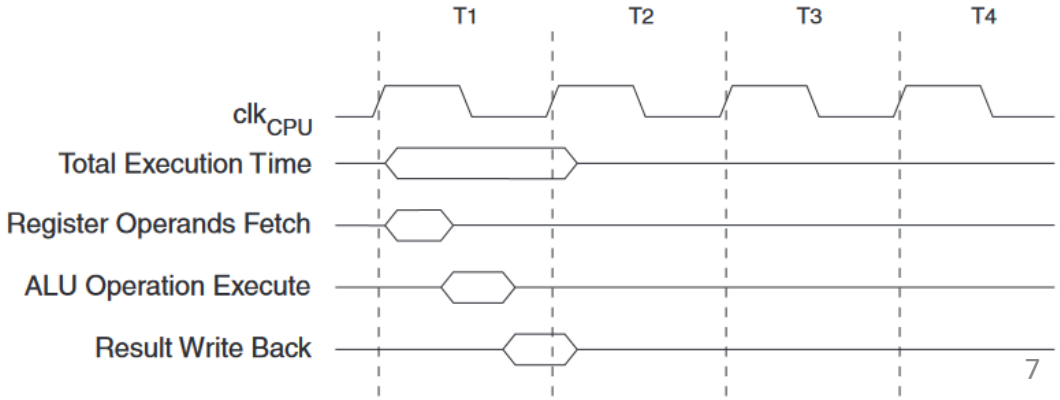
ATmega32 Features

- 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

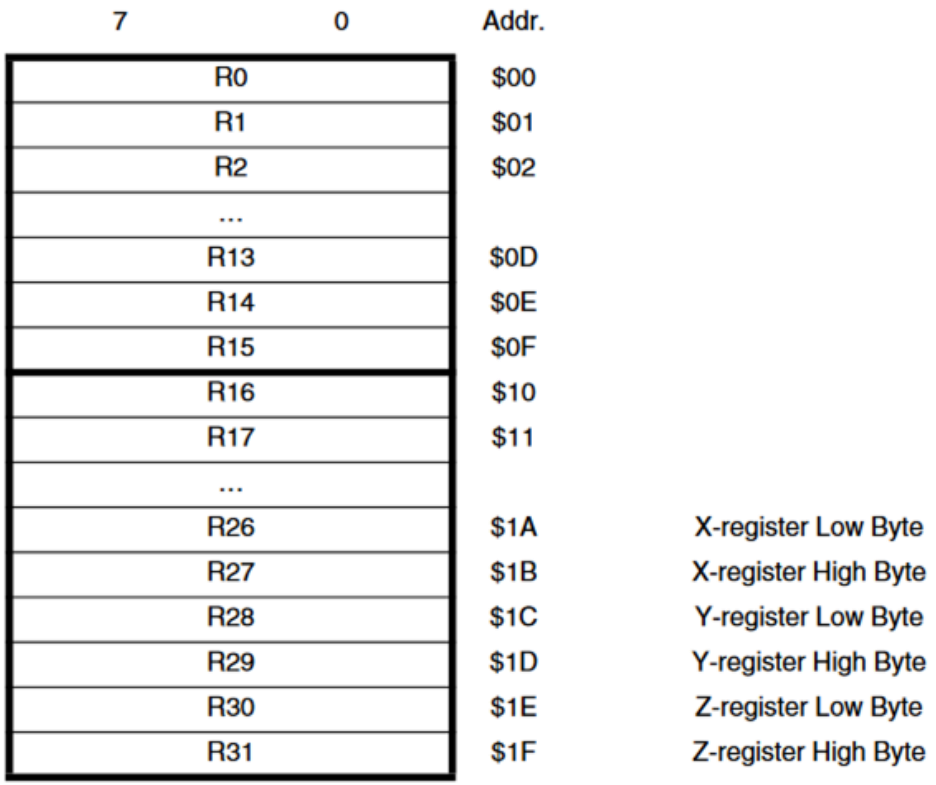
This concept enables instructions to be executed in every clock cycle



-
وابسته به معماری ATmega32 یکسری امکانات رو فراهم میاره این تراشه:
برای اینکه این میکروکنترلر حداکثر کارایی و یک مقدار موازی سازی در اختیار ما قرار بده از معماری هاروارد استفاده کرده ینی برای حافظه داده و برنامه باس مجزا و تراشه های حافظه مجزایی دیده شده
توی ATmega32 یک نوعی از پایپ لاین در اجرا شدن برنامه ها و دستورات می بینیم ینی زمانی که یک دستور داره اجرا میشه دستور بعدیش فچ میشه مثلا برای سیکل اولین دستور فچ میشه توی سیکل بعدی که تایم جلو می ره در عین حالی که دستور فعلی داره اجرا میشه سی پی یو میاد دستوری که بعد از این قراره اجرا بشه رو فچ می کنه و به همین ترتیب می ره جلو ینی همزمان با اینکه یک دستور داره اجرا میشه دستور بعدی هم میاد فچ میشه و اماده اجرا شدن میشه
معمولا این وابسته به ساختار اجرای دستورات هم است ینی اگر ما اگر زمان بندی هر کدوم از این سیکل ها رو ببینیم اپرند ها فچ میشن و توی یک سیکل ساعت همه اینا اتفاق می افته ینی اپرند ها فچ میشن و توی ALU عملی که قراره اجرا بشه صورت می گیره و بعد نوشته میشه توی یک دونه رجیستر و اونجایی که باید ذخیره بشه
این پاین لاینی که اینجا داریم خیلی کمک میکنه که یکسری موازی سازی هایی انجام بشه و کارایی ما بالا بره
نکته: اجرا شدن یک دستور همزمانه با فچ شدن دستور بعدی که این باعث میشه کارایی خیلی زیاد افزایش پیدا بکنه

Register File

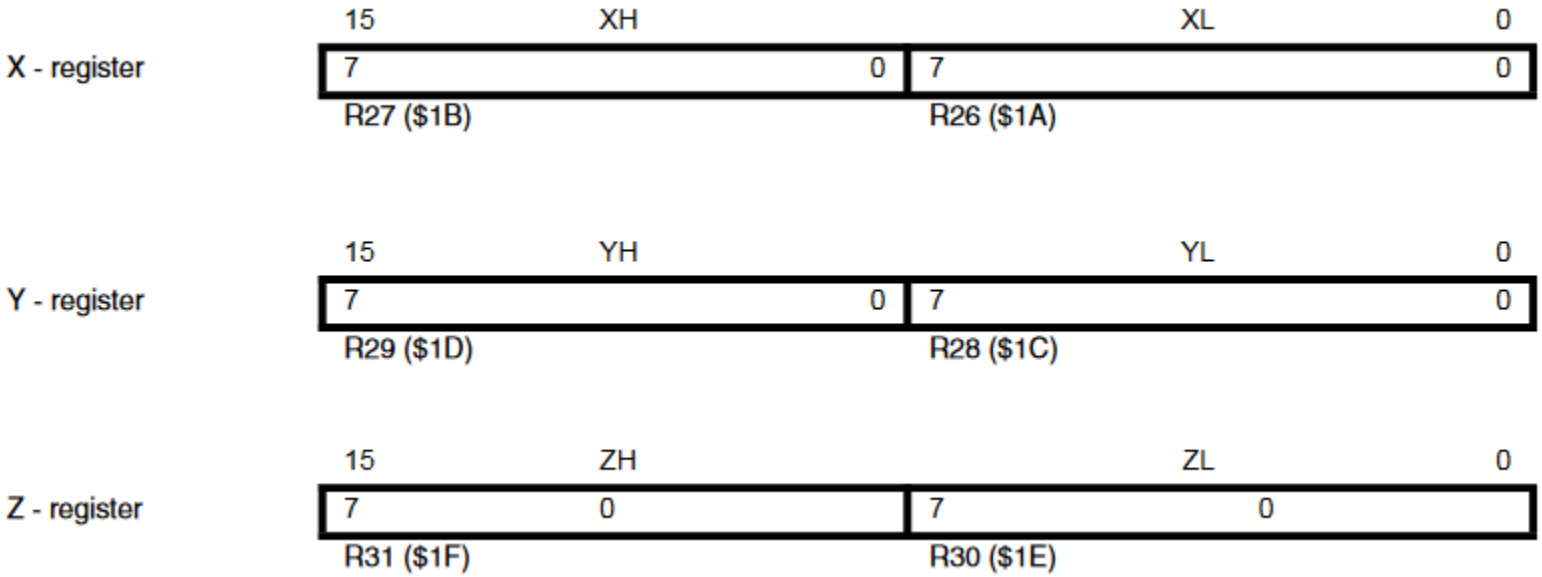
- Fast-access Register File
  - Contains 32 × 8-bit General Purpose working Registers (GPR)
  - ALU operates in direct connection with all the 32 GPR
  - with a single clock cycle access time
- Each register is assigned a data memory address
  - Mapping them directly into the first 32 locations of the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



-
ما 32 تا رجیستر 8 بیتی داریم ینی 32 تا رجیستر همه منظوره 8 بیتی داریم که از R0 تا R31 نامگذاری میشن و فضای ادرسی که به اینها اختصاص داده میشه از شروع فضای ادرس دهی ما هستش ینی از 00 تا 1F و تمام رجیسترها به صورت مستقیم به ALU متصل هستند و زمانی که برای دستیابی به این ها نیاز داریم یک سیکل ساعت است
فضای ادرس این ها بخشی از فضای SRAM ما است
بعضی از این رجیسترها کاربردهای خاصی دارند مثلا 1A , 1B با هم دیگه میان یک رجیستر تحت عنوان X تشکیل میدن و اونایی که توی شکل نشون داده شده رو ببین خودت
نکته: X , Y , Z سه تا رجیستر مجازی هستند

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



-
کاربرد این چند رجیستر اخر علاوه بر اون کاربردهای عمومی اینکه بتونیم به فضای حافظه در انواع مختلف ادرس دهی که اینجا وجود داره دسترسی داشته باشیم
در واقع دو تا رجیستر 8 بیتی که کنار هم است یک رجیستر 16 بیتی تشکیل میشه که می شه توی indirect addressing ازش استفاده بکنیم
اینا دو قسمت high , low دارند که جلوتر ازش می حرفیم

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	

-
استک یک فضایی از حافظه تلقی میشه که برای ذخیره کردن موقت داده ها و یا ادرس های بازگشتی که توی وقفه ها یا ساب روتینگ ها نیاز داریم به اون ها استفاده میشه
توی avr و ATmega32 استک ما بخشی از حافظه SRAM است ینی در واقع بالای اون قرار داره واستک پوینتر داره به خونه اول این اشاره میکنه
اتفاقی که میی افته اینه که چون استک بالای حافظه است با هر دفعه که یک چیزی داخل اون قرار بدیم این استک پوینتر کاهش پیدا میکنه ینی از سمت بالا به سمت پایین رشد خواهد کرد
توی اکثر سری های avr استک پوینتر با دوتا رجیستر 8 بیتی پیاده سازی میشه ینی قسمت low و high اش میشه این برای این است که بتونیم ادرس هایی با فواصل خیلی زیاد از اون کد برنامه ای که داریم بخوایم استفاده بکنیم اما توی سری هایی که حافظه اشون کمه معمولا از 8 بیت بیشتر استفاده نمیشه و همون 8 بیت هم کافی است

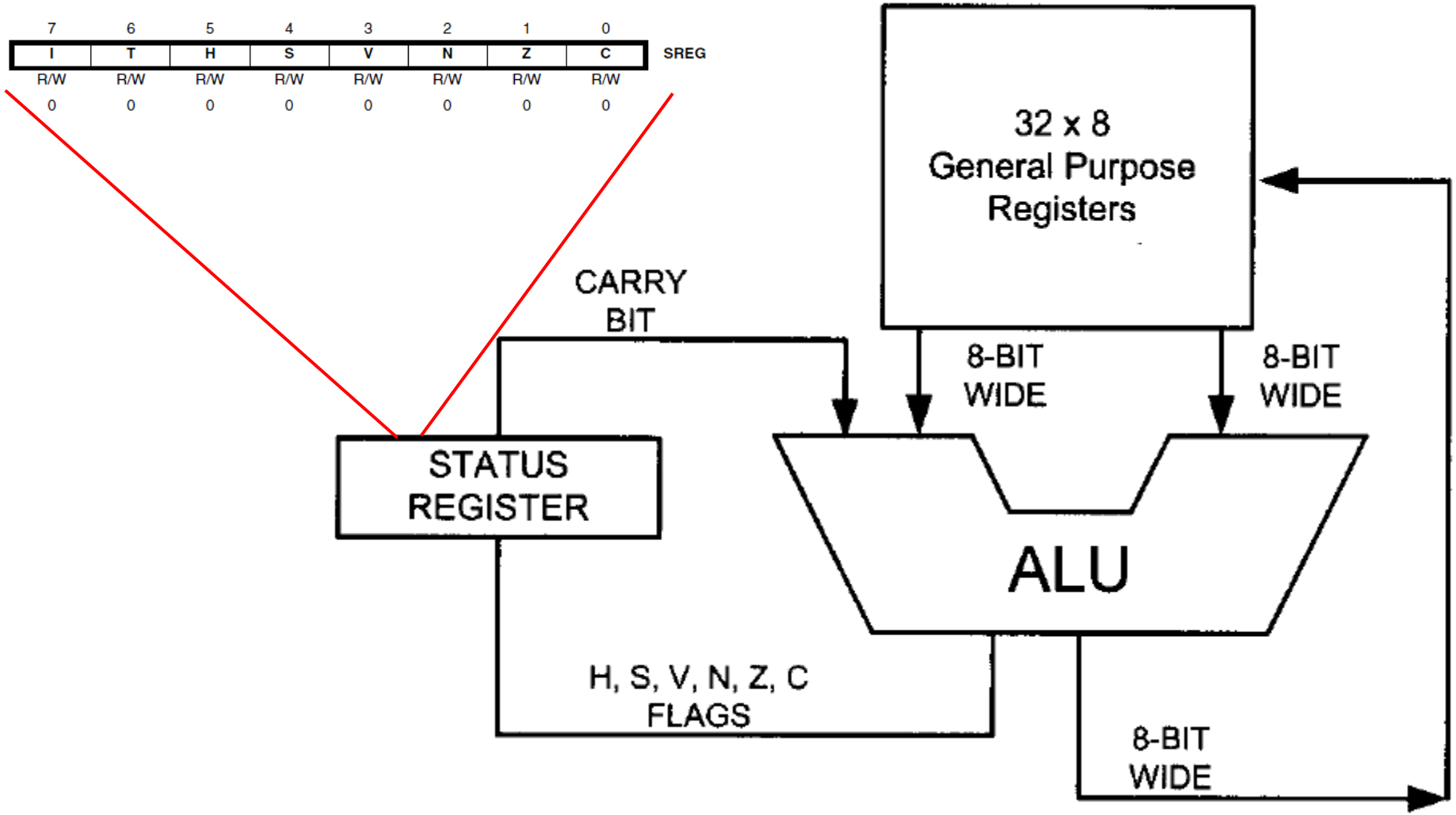
ALU

- 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	

-
عملیات هایی که ALU انجام میده می تونه بین رجیستر ها باشه یا می تونه بین یک رجیستر و یک مقدار ثابتی عملیات صورت بگیره - بعضا عملیات هایی داریم که فقط به یک رجیستر نیاز دارند معمولا یکسری دستوراتی داریم که اجرا شدن اینا می تونن توی سیستم تاثیرگذار باشه - یک رجیستری داریم تحت عنوان Status Register که وضعیت سیستم را به ما نشون میده ینی الان این اخرین عملیاتی که الان صورت گرفته چه تاثیری روی سیستم گذاشته - این ها یک سری پرچم ها هستند که توی بسیاری از عملیات ها واسه ما مهمه

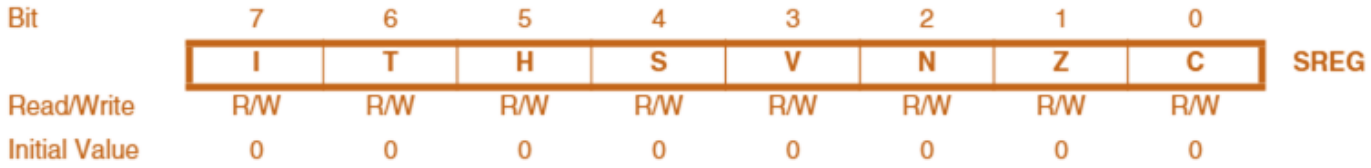
ALU


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

ساختار ALU:
32 تا رجیستر همه منظوره داریم که 8 بیتی هستند و مستقیما همشون به ALU متصل هستند و ALU اگر یک عملیاتی رو انجام میده و این روی نتیجه ما یک تاثیری داره و ما قراره اونو معطل بشیم از طریق یک سری پرچم هایی به ما اطلاع میده و این اطلاعات در یک رجیستر وضعیت قرار می گیره که این رجیستر 8 بیتی است و هر بیتش به یک منظوری مورد استفاده قرار میگیره

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	

رجیستر 8 بیتی وضعیت:
بیت 7:
برای مشخص کردن و فعال سازی وقفه است - حالا وقتی که یک وقفه می خواد صورت بگیره توی این سیستم ما یکسری وکتورهای وقفه هم داریم - پس سطح اولی که وقفه رو فعال میکنه بیت هفتم این رجیستر وضعیت هستش پس برای فعال سازی وقفه است این بیت
بیت 6:
متعلق به این است که یک فضای ذخیره سازی بیتی در اختیار ما قرار میده - مثلا اگر بخوایم از یک رجیستر یک بیت رو کپی بکنیم و از اون استفاده بکنیم می تونیم اون یک بیت رو منتقل بکنیم به اینجا
بیت 5:
زمانی که از نیمه اول 8 بیت یک کری داشته باشیم این فعال میشه
بیت 4:
بیت علامت
بیت 3:
زمانی است که یک سریزی رخ میده
بیت 2:
بیت منفی بودن
بیت 1:
بیت صفر بودن نتیجه اگر نتیجه ما صفر بشه توی اخرین عملیات این فعال میشه
بیت 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

-
ATmega32 یک سری دستوراتی داره که بتونیم اجرای برنامه رو کنترل بکنیم:
پرش های شرطی
پرش های غیر شرطی
کال هایی که می تونه روال برنامه رو از اون حالت ترتیبی که داره اجرا میشه خارج بکنه و توی بعضی از جاها خیلی کمک کننده باشه
یکسری دستورات دیگه هم داریم که وقتی که کال میکنیم می ریم داخل subroutine یا یک وقفه ینی روتین یک وقفه و اینجا نیاز داریم PC مون رو یک جا ذخیره بکنیم و PC توی استک ذخیره خواهد شد و توی بازگشت این ادرسی که قرار برگرده مستقیم می ره توی PC و از اونجا شروع میکنه اجرا شدن و یکسری محدودیت هایی داریم و اینه که استک ما نمی تونه از سایز SRAM مون در حد خیلی زیاد بیشتر بشه ینی در حالت کلی در سایز SRAM مون است استکمون
مدهایی ادرس دهی متفاوتی ATmega32 در اختیار ما قرار میده و دستورات avr عمدتا 16 بیتی هستند

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 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

1110	KKKK روی هم 4 بیت است	dddd روی هم 4 بیت است	KKKK روی هم 4 بیت است

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

-
دستورات زبان اسمبلی:
دستور لود است
با این دستور ما می تونیم یک ثابت رو ینی یک عدد 8 بیتی رو به یک رجیستر منتقل بکنیم
نکته:
1- اون عدد ثابت ما یک عدد 8 بیتی می تونه باشه
2- رجیستری هم که می تونیم اون مقدار رو به اون رجیستر منتقل بکنیم رجیستر های R16 تا R31 خواهد بود
توضیح دستور:
k رو توی رجیستر Rd می ریزه
رجیستر مقصد می تونه بین R16 تا R31 است
k هم می تونه بین صفر تا 255 باشه و k یک عدد 8 بیتی است
و بعد از اجرای این دستور PC یک واحد افزایش پیدا میکنه
این دستور یک دستور 16 بیتی هستش که اپکدش 1110 است توی جدول روبرو
یک رجیستر وضعیت داریم که هر عملیاتی که توی سیستم انجام میدیم اگر عملیاتی باشه که روی پرچم های مهم ما تاثیرگذار باشند مهمه که بدونیم برای دستور LDI هیچ اتفاقی برای پرچم ها توی این رجیستر وضعیت نخواهد افتاد ینی تاثیرگذار روی هیچکدوم از این پرچم ها نیست
این دستور یک word یا 2 بایت است و در یک سیکل ساعت هم اجرا میشه
نکته: بعد از اینکه دستور اجرا شد pc یک واحد زیاد میشه

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 ← Rr

	Syntax:	Operands:	Program Counter:
(i)	MOV Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← 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

-
دستور mov میاد مقدار یک رجیستر رو توی یک رجیستر دیگر کپی میکنه
رجیسترها می تونند هرکدوم از اون 32 رجیستر همه منظوره باشند
و بعد از االجرای این دستور PC یک واحد افزایش پیدا میکنه
یک دستور 16 بیتی است و اپکدش در اسلاید روبرو است
این دستور هیچ تاثیری روی رجیستر وضعیت نمیذاره
2 بایت یا یک word اپکد داره و در یک سیکل ساعت هم انجام میشه
نکته: بعد از اینکه دستور اجرا شد pc یک واحد زیاد میشه

ADD – Add without Carry

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

Operation:

(i) (i) Rd ← Rd + Rr

	Syntax:	Operands:	Program Counter:
(i)	ADD Rd,Rr	0 ≤ d ≤ 31, 0 ≤ r ≤ 31	PC ← 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

-
دستور جمع چندتا داریم که توی این نوع کری رو لحاظ نمیکنه
جمع میاد و مقادیر داخل دو رجیستر رو بدون در نظر گرفتن اون فلگ کری با هم جمع میکنه
حاصل جمع دوتا رجیستر رو در اخر می ریزه توی رجیستر اول
هر کدوم از این رجیستر ها می تونه یکی از اون 32 رجیستر همه منظوره باشه
و بعد از اجرای این دستور PC یک واحد افزایش پیدا میکنه
یک دستور 16 بیتی هستش
دستور جمع روی رجیستر وضعیت تاثیرگذار است به جز دوتا پرچم اول که مربوط به وقفه و فضای ذخیره سازی بیتی داریم روی 6 بیت دیگر رجیستر وضعیت تاثیرگذار است و اینکه چه تاثیری خواهد گذاشت وابسته به تعریف پرچم هاست که بعدا میگیم
یک word اپکد داره و توی یک سیکل ساعت هم اجرا میشه
نکته: بعد از اینکه دستور اجرا شد pc یک واحد زیاد میشه و برای بقیه دستورات هم به همین صورت است که در اسلاید بعدی گفته شده است

ADD – Add without Carry

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	⇔	⇔	⇔	⇔	⇔	⇔

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

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

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

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

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

Set if the result is $00; cleared otherwise.

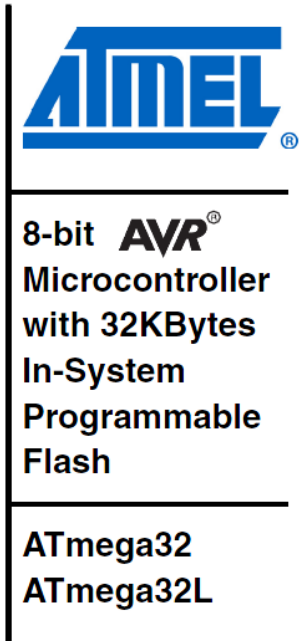
C $Rd7 \bullet Rr7 + Rr7 \bullet \overline{R7} + \overline{R7} \bullet Rd7$

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

چه تاثیری داره دستور جمع روی رجیستر وضعیت؟
دوتا رجیستر اول تحت تاثیر قرار نمی گیرند
رجیستر سوم که Half Carry هستش تحت تاثیر قرار میگیره ینی یک کری از نیمه اول مقدار به نیمه دوم منتقل میشه و این هم زمانی اتفاق می افته که شرطشو نوشته روبرو مثلا Rd3.Rr3 یک باشه یا یکی از اونا یک باشه
بیت علامت:
V سریز ما هستش که شرطشو توی اسلاید نوشته مثلا Rd7.Rr7.notR7 ینی بیت هفتم عملوندهای ما یک باشه و توی نتیجه ینی not R7 صفر باشه که تهش بشه یا یک یا اون شرط دیگر
بیت N یا Negative Flag که بیت پرارزشش یک باشه ینی R7 یک باشه
Z: طبق شرطش اگر همه بیت ها صفر باشه z فعال خواهد شد
C یا کری: زمانی اتفاق می افته که بیت های پرارزش اون ینی Rd7.Rr7 یک باشه یا بقیه شرایط

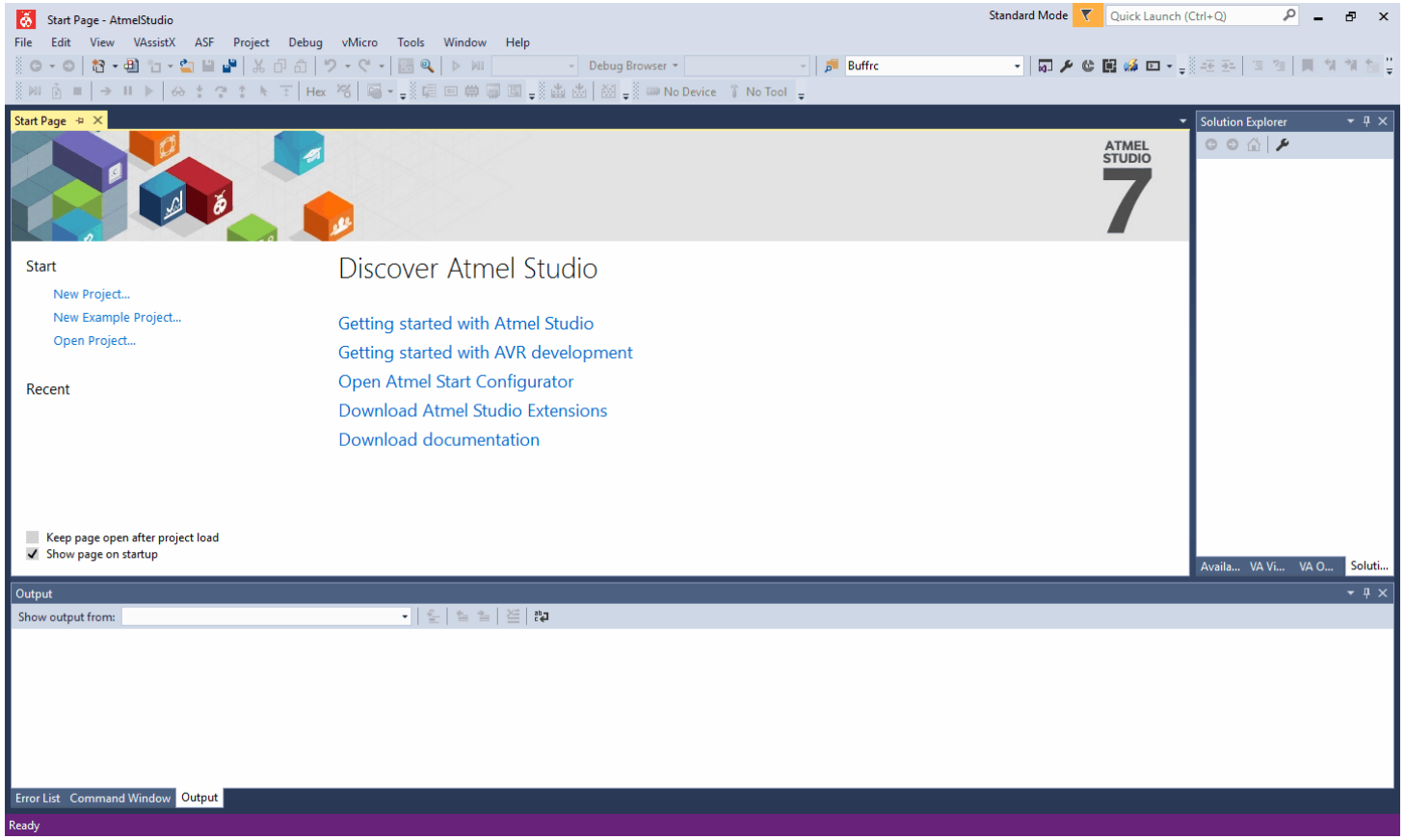
چرا اسمبلی؟!

- 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 Studio (Microchip Studio)



نصب نرم‌افزار و شروع به برنامه‌نویسی

پایان

موفق و پیروز باشید