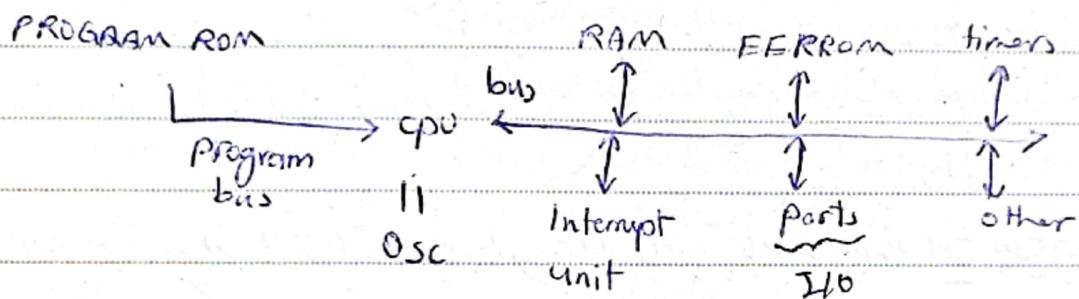


32bit AVR 32bit  
 60kb 8-bit (or 512KB 16bit CPU, 8-bit AVR)  
 (32KB 8-bit or 16KB 16bit)

classic - Special purpose - Tiny - Mega - ATtiny

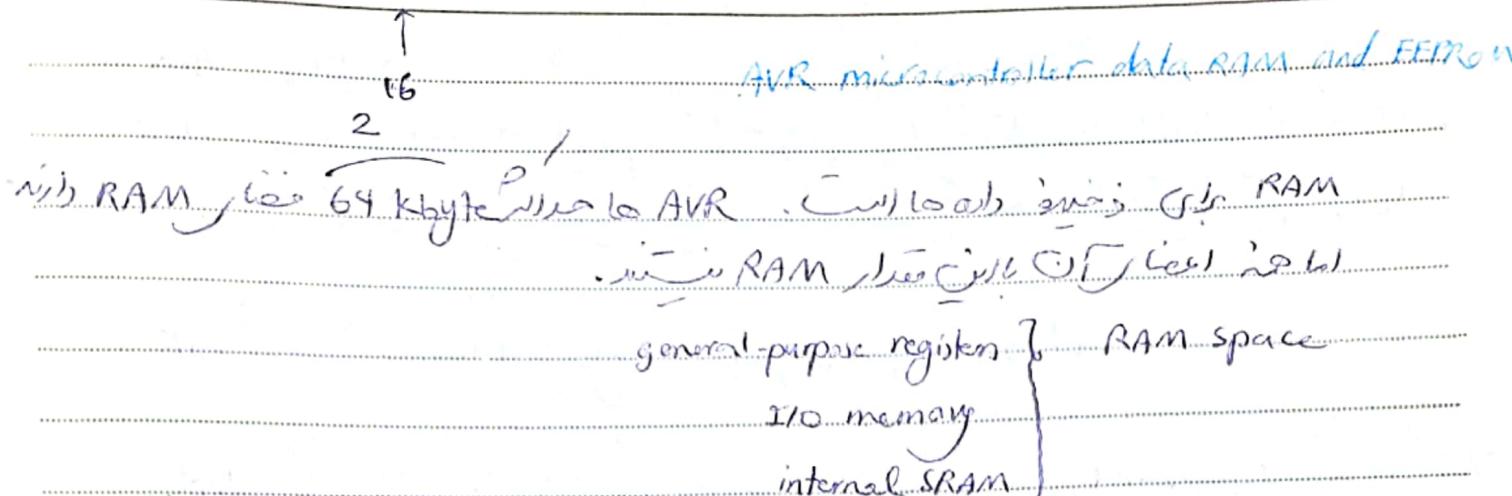
- In Harvard, has 1 CPU, single-chip AVR
- I/O ports, timer, data EEPROM, data RAM, ROM, on-chip program ROM
- Accumulator, PWM, ADC, Registers, etc. all to AVR
- USART, USB, CAN, I2C, SPI, USART



### AVR microcontroller program ROM

- 8M AVR has separate on-chip program ROM
- installed ROM is part of on-chip program ROM
- can have 1K to 256kbytes program ROM file
- Program storage is on-chip Flash memory

Ex 8 Question \*



I/O memory, SRAM is 64 kbytes, so is AVR has general-purpose regt 32  
 RAM - SRAM critical data is in SRAM  
 . critical data is in SRAM  
 critical data ← EEPROM

### AVR microcontroller I/O pins

مقدار این پین ها در AVR 86 تا 100 پین دارد

### peripherals

USART, and analog-to-digital ADC to AVR, and  
and 16-bit ADC and 10-bit as ADC  
and watchdog timer and 6 timer and two 16-bit timers  
com for serial port and AVR-based microcontroller UART  
CAN and VSB is for sub to SPI PC and other signals  
and USB

## AVR family overview

### Classic AVR (AT92Sxxxx)

• Customizable ROM size up to 128 Kbytes

### Mega AVR (ATmegaxxx)

• Customizable ROM size up to 120 Kbytes

- ↳ program memory → 4K to 256 K bytes
- ↳ package → 28 to 100 pins
- ↳ extensive peripheral set
- ↳ extended instruction set → rich instruction set

ATMega32 → ROM: 32K      ADC: 8      USART: 1  
 ↳ RAM: 2K      timers: 3  
 EEPROM: 1K      TQFP44  
 I/O pins: 32      PDIP40

### Tiny AVR (ATtinyxxxx)

• Customizable ROM size up to 128 Kbytes

- ↳ program memory: 1K to 8K bytes
- ↳ package: 8 to 28 pins
- ↳ limited peripheral set
- ↳ limited instruction set → Only multiply by 2 logic

Special purpose AVR

میکروکنترولر مخصوص ایندیکاتور و دیگر کارهای خاص IC  
با این قابلیت های بزرگتر از اینکه باشد  
برای ethernet و Zigbee و LCD و CAN و USB و ...  
و ... دارای پین PWM و FPGA

other microcontroller

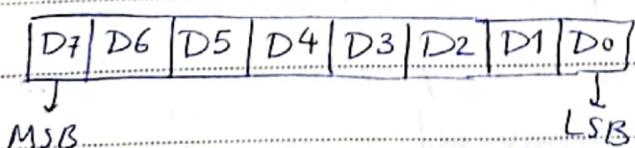
bus

Chapter 2 : AVR architecture and assembly programming

پردازنده های ایندیکاتور و دیگر کارهای خاص CPU

GPR ← general purpose register in AVR

پردازنده های ایندیکاتور و دیگر کارهای خاص CPU  
8-bit : پردازنده های ایندیکاتور و دیگر کارهای خاص CPU



R0 - R31 : پردازنده های 32-bit AVR

Subject :

Date .....

## LDI instruction

LDI Rd, K

register      immediate value

FF

Byte

1 1 1  
in 8 bits, we look at 8

values 00-FF in range 0-255 in 8 bits  
R31 R16 in range Rd  
R15 R0 (bytes)

hex  $\leftarrow$  0x  $\leftarrow \$$

dec  $\leftarrow$  nothing

binary  $\leftarrow$  0b

(01010101)15 in binary (F0)15 in hex  
FF / 255 in binary

figure 2.2  $\leftarrow$  in 8 bits

## Add instruction

Add Rd, Rr ; Add Rr to Rd and store the result in Rd

## AVR data memory

data mem space, code memory space : where we AVR

data  $\leftarrow$  data    code  $\leftarrow$  program

Subject :

Date .....

1 byte  
8-bit

\$0000	GPR	\$00 - \$1F $\leftarrow$ GPR in 32 byte
\$0020	SFR	(I/O registers) SFR in 64 byte
\$005F		
\$0060	SRAM	at least 8-bit $\downarrow$
⋮		
\$FFFF	TTT	serial communication, 64 status reg, ADC, I/O ports

\$0060 - \$00FF  $\leftarrow$  extended I/O memory

Internal data SRAM

scratch pad  $\leftarrow$   
(8-bit). Contains 64 bytes of memory.

SRAM vs EEPROM

non-volatile  $\leftarrow$  EEPROM

SRAM is volatile, loses data when power is lost.

RAM

$$\text{Data memory} = \text{I/O regs} + \text{SRAM} + \text{GPRs}$$

$$\text{ATmega32} = 64 + \frac{2048}{2K} + 32$$

Subject :

$0 \leq Rd \leq 31$  (load direct from data space) LDS Instruction

LDS Rd, K  
 $\uparrow \downarrow$   
 $\downarrow$

$$0000 \leq K \leq \$FFFF$   
 درست کردن

GPR

data mem

 $\uparrow \downarrow$  $\uparrow$ 

inj load ریسی داده ای، که از  
 درست GPR ریسی کرد R1 ریسی داده ای است  $K=0x1$  ریسی  
 - inj load Rd

(store direct to data space) STS Instruction

STS K, Rd → GPR

$$0000 \leq K \leq \$FFFF$

درست کردن (immediate) که  
 درست GPR ریسی کرد

LDI R16, 0x55

STS 0x230, R16

(W from I/O location) IN Instruction

 $0 \leq A \leq 63$  $\uparrow \downarrow$ 

IN Rd, A

درست I/O reg را در آن خواند و در  
 inj store Rd ریسی کرد

I/O register را در  $A=0x10$  ریسی کرد relative  $Rd=A$  ریسی

- 0x30 ← data.mem 0x55

درست I/O reg را در  $A=0x10$  ریسی کرد relative  $Rd=A$  ریسی

MINRO Rd, PINB

## IN VS. LDS

LDS : لديك عناصر IN ويجب ان تكتب LDS او IN ①

2 word 1 word لديك عناصر IN

IN او LDS او IN ②

لديك عناصر IN او LDS او IN ③

لديك عناصر IN او LDS او IN ④

لديك عناصر IN او LDS او IN ⑤

I/O او IN او perform data mem في LDS او IN

(I/O) او IN او perform data mem في LDS او IN

(I/O) او IN او perform data mem في LDS او IN

OUT A, Rr → 05RrS31

05A(S3)

مقدار عناصر (I/O, IN) او A او Rr او S31 او نحو ذلك

(OUT PORTD, R10)

immediate او port او عنوان او نحو ذلك او نحو ذلك او نحو ذلك او نحو ذلك

SRAM او I/O او نحو ذلك

(الحالات) \$5F او \$20 او I/O او نحو ذلك

immediate او 63 او 00 او relative او A او نحو ذلك

(LDI + STS / OUT)

او I/O او I/O او نحو ذلك

او GPR او نحو ذلك او نحو ذلك

IN R0, PIND

OUT PORTA, R0

PIND → PORTA

**Subject :**

Date .....

## Mov Instruction

Mov Rd, Rr

Mov R10, R20  $\rightarrow R10 = R20$

GPR INC Rd

## INC Instruction

$$\text{INC R2} \Rightarrow R2 = R2 + 1$$

\* متدرب اوس خاص (برف GPR) ران توان مقسم نماینگری دارد و  
با دلخواه حجم بزرگ، حجم نماینگری نیم و از STS استفاده نمی‌کند

LDS → INC → STS

## SUB Instruction

SUB  $R_d, R_r$  :  $R_d = R_d - R_r$

GPR

## DEC Instruction

DEC Rd ; Rd = Rd - 1

GPR

COM Rd

COM Instruction

LDI R16, 0x55

COM R16

$$\rightarrow R16 = 0x55$$

LDI R10, 0x55

OUT PORTB, R10

Loop: COM R10

OUT PORTB, R10

JMP Loop

## AVR Status register

SREG	I	T	H	S	V	N	Z	C
global interrupt enable	bit copy	half carry	sign flag	overflow flag	zero flag	negative flag	carry flag	

I10 neg  $\leftarrow$  (SREG) Cw 8Carry flag  $\leftarrow$  D7Sign flag  $\leftarrow$  D6Signed  $\leftarrow$  [ LittleEndian format ]LittleEndian format  $\leftarrow$  N VN V  $\leftarrow$  S(W: AL) D4 ~ D3 ; carry  $\leftarrow$  H

Subject :

Date .....

flag bits and decision  
making

- uses of fist (branch) of flags in programs

BRLO → C = 1

BRSH → C = 0

BREQ → Z = 1

BRNE → Z = 0

BRMI → N = 1

BRPL → N = 0

BRVS → V = 1

BRVC → V = 0

→ AVR

in AVR program . there are 8 data types -

\$ / 0x ← hex  
nothing ← dec  
0B / 0b ← binary  
single quote ← ASCII  
" " ← if string → double quote  
· DB

Assembler directives

into CPU instructions . Add , LD , etc . pseudo instructions ←

into a directive . device . ORG . EQU etc

they are used to tell the assembler what to do

equate ← .EQU

بلکه تعریف یک constant است این استفاده محدود به حافظه شخصی نشود و مقدار را باید با عنوان یعنی label associate کنید و مقدار را در آن label استفاده کنید مثلاً مربوط شده با جایزه label می‌شود. این استفاده از EQU بزرگ داده شده باشد زیرا داشت طبق این اس تا کم در مرور کردن بحث داشته باشد و پس از آن را در label جستجو کنید و از search بحث داشته باشد. استفاده ایم دستور I/O بجز اینها مخصوصاً با همین دلیل ایجاد شده است.

.EQU PORTB = 0x1B

.SET

.SET .EQU است، تفاوت این دو در مقدار شده توسط assign است. این دوباره شد.

برای SFR دو نوع relative OUT، IN و INOUT دارد

از دستور DB بلکه حافظه Code ROM برای دستورات می‌شود

text

Actual bus relative .inout assign SFR ← .EQU لذت

برای دستورات SFR دو نوع assign دارد بحث داشته باشد .EQU شم خوبی برای internal SRAM داشته باشد

• ORG (origin)

• ایندیکت ایجاد کردن آدرسی برای آغاز کد

• INCLUDE

(#include "file") در اینجا باید مسیر

• INCLUDE "M32DEF.INC" ← ATmega32 پر نمود

لایبل را در اینجا

alphabet + digits + ? + . + @ + ( ) + \$

و همچنانه تابعی که لایبل را در اینجا

(LDI, ADD etc) را با آن نمایش دهد

abbreviation ← mnemonic?

نیز باید اینجا

که low-level CPU را داشتی

امثلی → [label:] mnemonic [operand] [; comment]

HERE : JMP HERE ; stay here forever 1 or 2

که اینجا یعنی یک لایبل را پس از آن داشتی \*

low-level ← که باید CPU (جگله) را بپرسید که آنرا

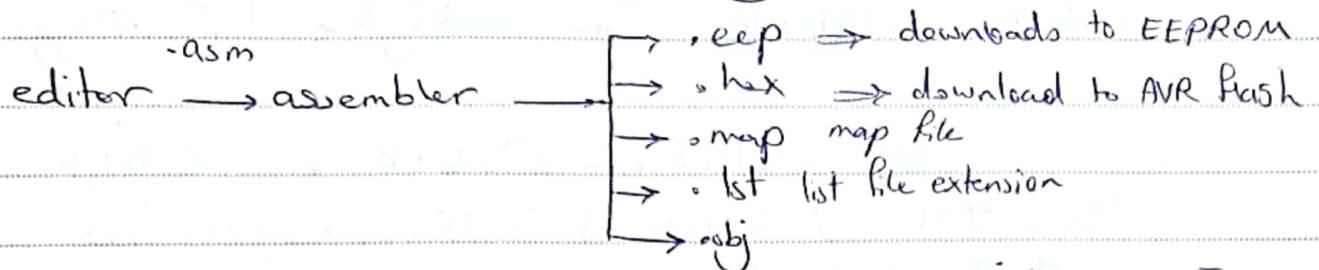
که اینجا یعنی یک لایبل را پس از آن داشتی ①

که اینجا یعنی یک لایبل را پس از آن داشتی ②

\$67      \$55

نحوه اجراء نا اسفل

برازشیون و استفاده از Text editor



source file ← asm file  
 emulator / Simulator ← obj file

label ← map file

EQU SUM 00000300

CSEG HERE 00000300

ناتیجہ کے حسب حفظ کرنا اور اسے save کرنا۔ اسے optional کہا جاتا ہے۔

کام کرنے والے سیگنال کو source code کے شکل میں دستیاب کرنا۔

کام کرنے والے سیگنال کو in

نامہ کے لئے لینے والے سیگنال کو user کے لیے لینے والے سیگنال کو NotePadAV.

program Counter

کام کرنے والے سیگنال کو PC کو AVR کو کھینچنے کا کام کرنے والے سیگنال کو fetch، program ROM کو opcode، CPU کی کام کرنے والے سیگنال کو

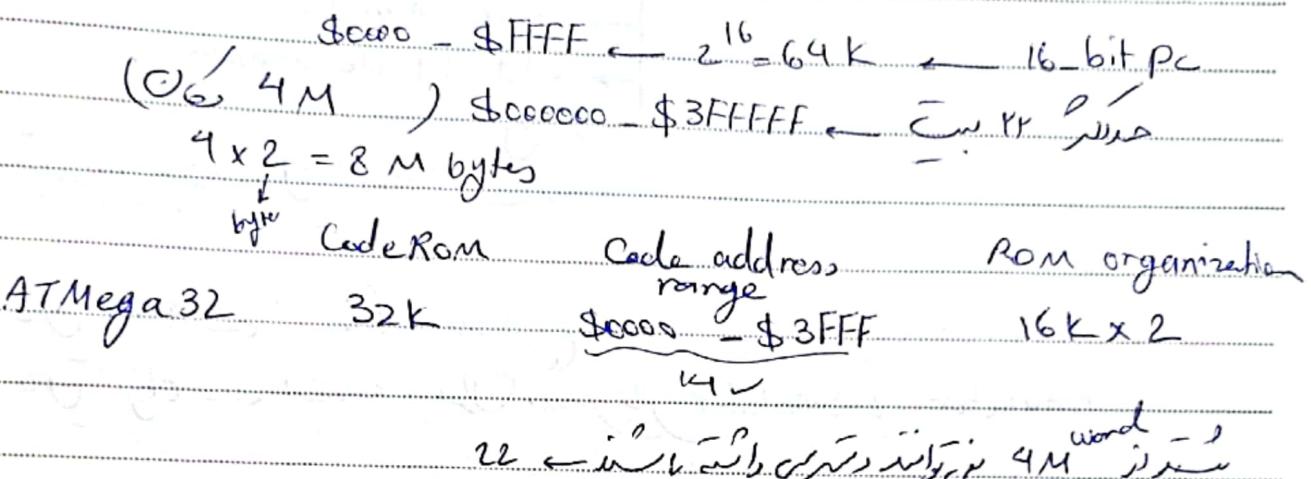
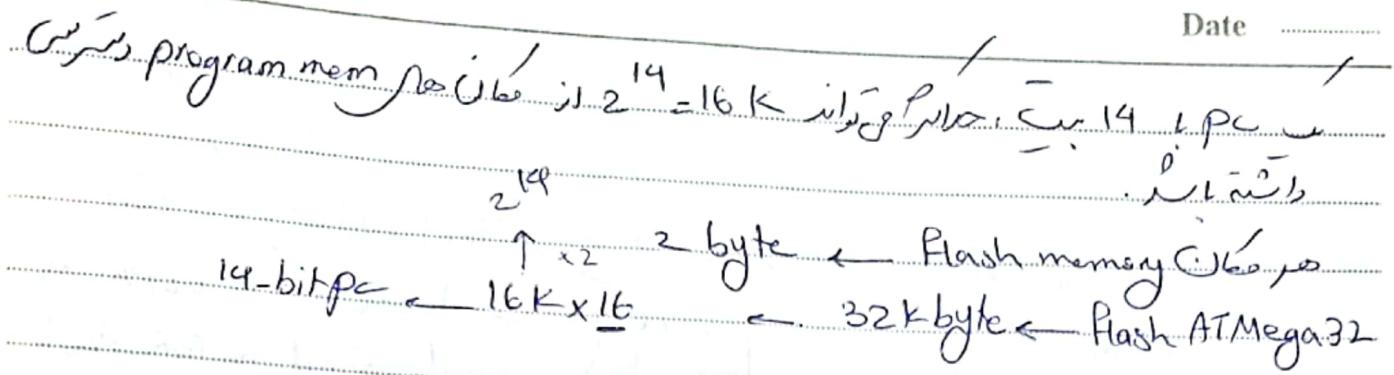
زندگی کے لئے

MICRO

cpu کا کام کرنے والے سیگنال ← ↑ program Counter

Subject :

Date \_\_\_\_\_



### Program ROM u AVR

start  $\leftarrow$  program word 0000 word  $\leftarrow$  AVR  
new operand, opcode  $\leftarrow$  AVR in ROM  $\leftarrow$  new word  
writing word 0000 word in ROM

1110	KKKK	ddddd	KKKK
------	------	-------	------

LDI Rd, K  $16 \leq d \leq 31, 0 \leq K \leq 255$   
 $\frac{2^5 - 1}{4\text{bit}}$   $\frac{2^8 - 1}{8\text{bit}}$

R16 = 0

R17 = 1

LDI R16, 0x25

E	2	0	5
---	---	---	---

R18 = 2

R31 = 15

0000	11rd	dddd	rrrr
------	------	------	------

Add Rd, Rr       $0 \leq d \leq 31$        $0 \leq r \leq 31$

$10000$        $10001$

$2^5 - 1$

Add R16, R17       $\rightarrow$   $\frac{0000}{0} \frac{11ff}{F} \frac{0000}{0} \frac{0001}{1}$

$2^9 - 1$

1 word = 2 bytes  
= 16 bits

يتكون من 2 بิต في الموضع 4 ، STS ، JMP ، دستورات  
1 word / 2 word

من بين دستورات 2 أweis حافظة راسمال من موضع

حقل آدرس 0A ، 09 حافظة على

LDI 0000

Add 0001

LDI 0002

:

JMP 0009

000A

### ROM width AVR

(address line) يوصل لـ 16 بت ، أو 2 بيت (1 word) ، 2 بيت (2 words)  
0000-FFFF 64K بـ 16 بت ، 64K بـ 32 بت ، 32 بت ، 64K بـ 64 بت ، 32 بت ، 64K بـ 128 بت ، 64K بـ 256 بت ، 128 بت ، 256 بت ، 512 بت ، 1024 بت ، 2048 بت ، 4096 بت ، 8192 بت ، 16384 بت ، 32768 بت ، 65536 بت ، 131072 بت ، 262144 بت ، 524288 بت ، 1048576 بت ، 2097152 بت ، 4194304 بت ، 8388608 بت ، 16777216 بت ، 33554432 بت ، 67108864 بت ، 134217728 بت ، 268435456 بت ، 536870912 بت ، 1073741824 بت ، 2147483648 بت ، 4294967296 بت ، 8589934592 بت ، 17179869184 بت ، 34359738368 بت ، 68719476736 بت ، 137438953472 بت ، 274877906944 بت ، 549755813888 بت ، 1099511627776 بت ، 2199023255552 بت ، 4398046511104 بت ، 8796093022208 بت ، 17592186044416 بت ، 35184372088832 بت ، 70368744177664 بت ، 140737488355328 بت ، 281474976710656 بت ، 562949953421312 بت ، 1125899906842640 بت ، 2251799813685280 بت ، 4503599627370560 بت ، 9007199254741120 بت ، 18014398509482240 بت ، 36028797018964480 بت ، 72057594037928960 بت ، 144115188075857920 بت ، 288230376151715840 بت ، 576460752303431680 بت ، 115292150460686320 بت ، 230584300921372640 بت ، 461168601842745280 بت ، 922337203685490560 بت ، 1844674407370981120 بت ، 3689348814741962240 بت ، 7378697629483924480 بت ، 14757395258967848960 بت ، 29514790517935697920 بت ، 59029581035871395840 بت ، 118059162071742791680 بت ، 236118324143485583360 بت ، 472236648286971166720 بت ، 944473296573942333440 بت ، 1888946593147884666880 بت ، 3777893186295769333760 بت ، 7555786372591538667520 بت ، 1511157274582307735040 بت ، 3022314549164615470080 بت ، 6044629098329230940160 بت ، 12089258196658461880320 بت ، 24178516393316923760640 بت ، 48357032786633847521280 بت ، 96714065573267695042560 بت ، 193428131146535390085120 بت ، 386856262293070780160240 بت ، 773712524586141560320480 بت ، 1547425049172283120640960 بت ، 3094850098344566241281920 بت ، 6189700196689132482563840 بت ، 12379400393378264965127680 بت ، 24758800786756529930255360 بت ، 49517601573513059860510720 بت ، 99035203147026119721021440 بت ، 198070406294052239442042880 بت ، 396140812588104478884085760 بت ، 792281625176208957768171520 بت ، 1584563250352417915536343040 بت ، 3169126500704835831072686080 بت ، 6338253001409671662145372160 بت ، 1267650600281934332428744320 بت ، 2535301200563868664857488640 بت ، 5070602401127737329714977280 بت ، 10141204802455474659429554560 بت ، 20282409604910949318859109120 بت ، 40564819209821898637718218240 بت ، 81129638419643797275436436480 بت ، 162259276839287954550872873920 بت ، 324518553678575909101745747840 بت ، 649037107357151818203491495680 بت ، 1298074214714303636406982991360 بت ، 2596148429428607272813965982720 بت ، 5192296858857214545627931965440 بت ، 10384593717714429091255863930880 بت ، 20769187435428858182511727861760 بت ، 41538374870857716365023455723520 بت ، 83076749741715432730046911447040 بت ، 166153499483430865460093822894080 بت ، 332306998966861730920187645788160 بت ، 664613997933723461840375291576320 بت ، 1329227995867446923680750583152640 بت ، 2658455991734893847361501166305280 بت ، 5316911983469787694723002332610560 بت ، 1063382396693957538944600466521120 بت ، 2126764793387915077889200933042240 بت ، 4253529586775830155778401866084480 بت ، 8507059173551660311556803732168960 بت ، 17014118347078306223113607464337920 بت ، 34028236694156612446227214928675840 بت ، 68056473388313224892454429857351680 بت ، 13611294677662644978490855971470320 بت ، 27222589355325289956981711942940640 بت ، 54445178710650579913963423885881280 بت ، 10889035742130155982792684777776320 بت ، 21778071484260311965585369555552640 بت ، 43556142968520623931170739111105280 بت ، 87112285937041247862341478222210560 بت ، 174224571874082455724682956444421120 بت ، 348449143748164911449365912888842240 بت ، 696898287496329822898731825777684480 بت ، 139379657499265964579746365155336960 بت ، 278759314998531929159492730306673920 بت ، 557518629997063858318985460613347840 بت ، 1115037259985127716637973921226755680 بت ، 2230074519970255433275947842453511360 بت ، 4460149039940510866551895684907022720 بت ، 8920298079881021733103791369814045440 بت ، 1784059615976204346620758273962808880 بت ، 3568119231952408693241516547925617760 بت ، 7136238463884817386483033095851235520 بت ، 1427247692776963477296606619170467040 بت ، 2854495385553926954593213238340934080 بت ، 5708990771107853909186426476681868160 بت ، 1141798154221570781837245295336376320 بت ، 2283596308443141563674490590672752640 بت ، 4567192616886283127348981181345505280 بت ، 9134385233772566254697962362690505560 بت ، 18268770467545132509395924725381011120 بت ، 365375409350902650187918494507620222240 بت ، 730750818701805300375836989015240444480 بت ، 146150163740361060075167977803048088960 بت ، 292300327480722120150335955606096177920 بت ، 584600654961444240300671911212192355840 بت ، 1169201309922884480601343822424384711680 بت ، 2338402619845768961202687644848769423360 بت ، 4676805239691537922405375289697538846720 بت ، 9353610479383075844810750579395077693440 بت ، 18707220958766151689621501158790155386880 بت ، 37414441917532303379243002317580310737760 بت ، 74828883835064606758486004635160621475520 بت ، 14965776767012921351692009267532124291040 بت ، 29931553534025842703384008535064248582080 بت ، 59863107068051685406768017070128497164160 بت ، 119726214136103370813536034140256994328320 بت ، 239452428272206741627072068280513988656640 بت ، 478904856544413483254144013561027977313280 بت ، 957809713088826966508288027122055954626560 بت ، 191561942617615393301657605424411188933120 بت ، 38312388523523078660331521084882237766240 بت ، 76624777047046157320663042169764475532480 بت ، 153249554094092314641326084339528951064960 بت ، 306499108188184629282652168678557802129920 بت ، 612998216376369258565304337357115604259840 بت ، 122599643275273851713060867471423120919760 بت ، 245199286550547703426121734942846241839520 بت ، 490398573101095406852243469885692483679040 بت ، 980797146202190813704486939771384967358080 بت ، 196159429240438162740897387954276993476160 بت ، 392318858480876325481794775908553986952320 بت ، 784637716961752650963589551817107973874640 بت ، 1569275433923505301927179103634215947749280 بت ، 3138550867847010603854358207268431894988560 بت ، 6277101735694021207708716414536863789777120 بت ، 1255420347138804241541743282907372757554240 بت ، 2510840694277608483083486565814745515108480 بت ، 5021681388555216966166973131629491030216960 بت ، 10043362777110439332339466633248922060433920 بت ، 20086725554220878664678933266497844120867840 بت ، 40173451108441757329357866532995688241735680 بت ، 80346902216883514658715733065985376483471360 بت ، 16069380443376702917543566613197155296682720 بت ، 32138760886753405835087133226394310593365440 بت ، 64277521773506811670174266452788621186730880 بت ، 128555043547013623340348532905577242373461760 بت ، 257110087094027246680697065811154484746923520 بت ، 51422017418805449336139413162230896949384640 بت ، 10284403483761089867227882632446179389769280 بت ، 20568806967522179734455765264892358779538560 بت ، 41137613935044359468911530529784717558777120 بت ، 82275227870088718937823061059569435117554240 بت ، 16455045574017743787564632211913887023510880 بت ، 32910091148035487575129264423827754047021760 بت ، 65820182296070975150258528847655508094043520 بت ، 131640364592141950300517057695311016188087040 بت ، 263280729184283900601034115390622032376174080 بت ، 526561458368567801202068230781244064752348160 بت ، 105312291673713560240413646156248812904696320 بت ، 210624583347427120480827292312497625809392640 بت ، 421249166694854240961654584624995251618785280 بت ، 842498333389708481923273169249985503237570560 بت ، 168499666677941696384656338489971006465141120 بت ، 336999333355883392769312676979942012930282240 بت ، 67399866671176678553862535395988402586156480 بت ، 134799733342353357107312670791976805172312960 بت ، 26959946668470671421462534158395361034465920 بت ، 53919893336941342842925068316786722068931840 بت ، 107839786679882685685850136635774441137863280 بت ، 215679573359765371371700273271548882275726560 بت ، 431359146719530742743400546543097764551453120 بت ، 86271829343856148548680019308619553902906240 بت ، 172543658687712297097360038617239107805812480 بت ، 345087317375424594194720077234478215611624960 بت ، 690174634750849188389440154468956431223249920 بت ، 138034926910169837677880308893791286446499920 بت ، 27606985382033967535576061778758257289299840 بت ، 55213970764067935071152013557516514578599680 بت ، 110427941528135870142304027115033029157199360 بت ، 220855883056271740284608054230066058314398720 بت ، 441711766112543480569216010460132116628797440 بت ، 883423532225086961138432020920264233257594880 بت ، 1766847064450173922676640418405284466155897760 بت ، 353369412890034784535328083681056893231179520 بت ، 706738825780069569070656167362113786462359040 بت ، 141347765156013913814131233472426557292478080 بت ، 282695530312027827628262466944853114584956160 بت ، 565391060624055655256524933889706229169912320 بت ، 113078212124811131051304966777941245839824640 بت ، 226156424249622262102609933555882491679649280 بت ، 452312848499244524205219867111764983359298560 بت ، 90462569699848904841043973422352976678597120 بت ، 180925139399697809682087946844705953357194240 بت ، 361850278799395619364175893689411906714388480 بت ، 723700557598791238728351787378823813428776960 بت ، 1447401115197582477456703574757647626857553920 بت ، 289480223039516495491340714951529531371511840 بت ، 578960446079032990982681429903058562743023680 بت ، 115792089215806598196536285980611712546047360 بت ، 231584178431613196393072571961223425092094720 بت ، 463168356863226392786145143922446850184189440 بت ، 926336713726452785572290287844893700368378880 بت ، 1852673427452905571144580575689787400736757760 بت ، 3705346854905811142289161151379575801473515520 بت ، 741069370981162228457832230275915160294703040 بت ، 148213874196232445691566446055183032058852080 بت ، 296427748392464891383132892110366064117704160 بت ، 592855496784929782766265784220732128235408320 بت ، 1185710993569859565532531568441464256470176640 بت ، 2371421987139719131065063136882928512940353280 بت ، 4742843974279438262130126273765857025880706560 بت ، 9485687948558876524260252547531714051761413120 بت ، 1897137589711775304852050509506342810352282640 بت ، 3794275179423550609654101019012685620704565280 بت ، 7588550358847101219308202038025371241409130560 بت ، 1517710071769420243861640407605074248281821120 بت ، 3035420143538840487723280815210148496563642240 بت ، 6070840287077680975446561630420298993287284480 بت ، 1214168057415236195089123266084058798657456960 بت ، 2428336114830472390178246532168175597354913920 بت ، 4856672229660944780356493064336351194709827840 بت ، 9713344459321889560712986128672672389419657680 بت ، 19426688918643779121425872257345346778839355360 بت ، 38853377837287558242851744514690693557678710720 بت ، 77706755674575116485703589029381387115357421440 بت ، 155413511349150232911407780487726774230714842880 بت ، 310827022698300465822815560975453548461428855760 بت ، 621654045396600931645631121950907096922857111520 بت ، 124329809079200186329266224390181419385574223040 بت ، 248659618158400372658532448780362838771148446080 بت ، 4973192363168007453170648

Harvard Architecture

، AVR

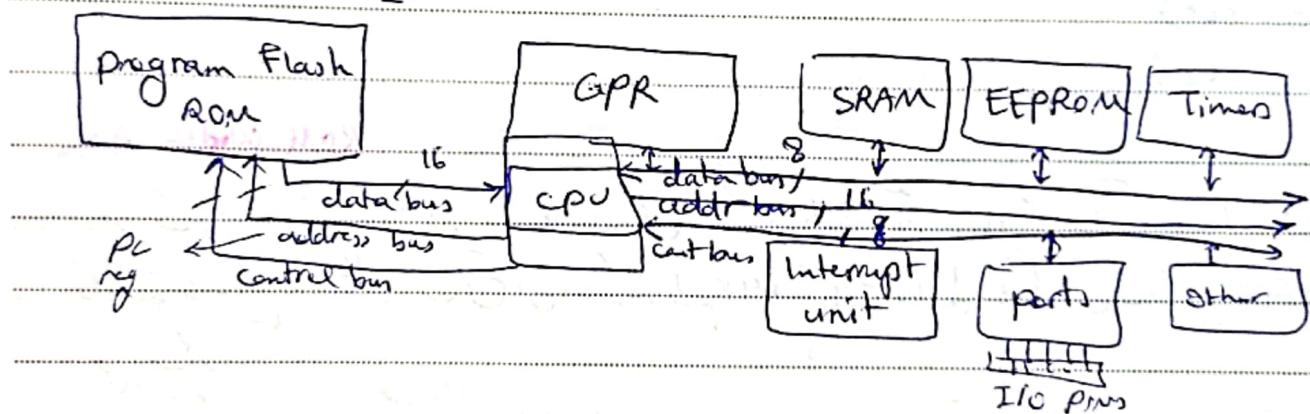
جهاز data bus ، program Flash ROM ~ جهاز program bus  
جهاز CPU ~ data bus

جهاز address bus ، (جهاز 1/جهاز 2) ~ 16 ← program bus  
flash ROM جهاز نسبت CPU ~ لـ CPU PC reg

جهاز data bus ، (جهاز 1/جهاز 2) ~ 8 ← Data bus

جهاز address bus . جهاز SRAM

جهاز 64 K  $\frac{1}{2^{16}}$  بـ data memory



(جهاز 2) ~ 16 ، ROM جهاز 1 (جهاز 1) ~ 8 (SRAM) RAM جهاز \*

data bus جهاز address bus ~ 0x90 ، LDS R20, 0x90

جهاز GPR ، جهاز data bus جهاز 8 بـ SRAM جهاز

## Little-endian vs. big-endian

high word, high-byte, endianness low word, low-byte. ROM is  
 (jkl) little-endian ↪

## LDI instruction Formation

LDI Rd, K 16 bits &lt; 31 0 ≤ K ≤ 255

2-byte ↪

1110	KKKK	dddd	KKKK
------	------	------	------

## Add instruction formation

Add Rd, Rr

opcode	0000	11rd	dddl	rrrr
0131 < 31	0000	1101	0000	0000

2-byte ↪

## STS instruction Formation

Opcode ← Clk → 16

2 byte

STS K, Rr

1001	001r	rrrr	0000
KKKK	KKKK	KKKK	KKKK

→ \$0000 - \$FFFF

Subject :

Date .....

### LDS instruction Formation

LDS Rd, K

$0 \leq d \leq 31$

$0 \leq K \leq 65535$

4-byte ↪

1001	000d	dddd	0000
KKKK	KKKK	KKKK	KKKK

### IN instruction Formation

IN Rd, A

$0 \leq d \leq 31$

$0 \leq A \leq 63$

opcode

2-byte ↪

1011	0AAd	dddd	AAAA
------	------	------	------

### Out instruction Formation

Out A, Rd

2-byte ↪

1011	1AAd	dddd	AAAA
------	------	------	------

### JMP instruction Formation

JMP K

$0 \leq K \leq 4M$

4-byte ↪

opcode ↪ Cw 10

1001	010K	KKKK	110K
KKKK	KKKK	KKKK	KKKK

RAM / is for address K

## RISC Architecture in AVR

- سروره افزاسی مدرن CPU
- ① باید برین فرکانس clock (مادولان) معنی وارد خواهد شد
- باید درسته داشت hand-held
- ② معاشر دارای این استعداد bus ها را با آرین اطمینان نداشت
- CPU = (data, code)
- RISC CPU دارای این استعداد از معاشر

رزص در رسم این استعداد است.

### reduced instruction set computer RISC

#### complex instruction set computer CISC

قدرت زیاد دستورات رئیسیت بین آنها باشد، پر کردن برای این سیستم دشوار است

ویژگی های رسی

- CISC دارای Instruction های کمتر از RISC است. اما در CISC دستورات عوتانه 1-2-3 بایت باشند. این سبب خود را که در CISC دستورات کوتاه دستورات را در چند بایت معلوم نمایند.
- دستورات دستورات بیش از 4 بایت هستند.
- دستورات بیش از 4 بایت هستند.
- دستورات بیش از 4 بایت هستند.

32 bit RISC دارای 32 bit GPR هاست.

- حجتی دارد. از 32 bit فکر نماید که کمتر جدا تغییر داده شود. می تواند از 32 bit استفاده کرد.
- از خوبی های دستورات RISC این است که stack بزرگ را نداشته باشد.
- نیازی ندارد این صورت را در CISC داشت.

۴) سیمین درس RISC این است که ۱/۵ درستورات در ۳۰۰ سایل اجرا می شوند.  
برای سیسی CISC این ۱/۵ درستورات در کل کل سایل طول اجرا کشیده هم می شوند.  
Compiler در کل کل سایل لیبل را می بیند. این طریق چند code scheduling است.

Ques. RISC. If code, data by large word, RISC (a) consists of F opcode (r data) or (r data) (i : 2),

Microinstruction بدلیل اینکه بیان را در میان کامپیوٹرها CISC (4  
RISC و MIPS) میانجیگردانی نمایند. اینکه پس از ۴۰-۶۰٪  
hardware باشد.

Microprocessor design - Load/Store architecture

ـ مجموعات حجمها يزيد على ٥٠٠ متر  $\leftarrow$  simulator

## \* Chapter 3 : Branch, Call, and time delay loop \*

و<sup>و</sup> دستور loop دستوری است که یک بلوک کد را تکرار می کند. درین دستور از دستور for متفاوت است زیرا در دستور loop باید تعداد دوره های تکرار را مشخص نمود.

BRNE

branch if not equal ↵

branch if zero flag is set - استفاضة if zero flag is  
راهنمایی کنید - در اینجا zero flag  $\neq 0$  بود - پس را لیکن هر دو صفر نبود

Back: \_\_\_\_\_

Def-Rn

Original BRNE Back Z=1

jet (W) land - clear  
Sep 2011 255 atm

loop inside a loop

بیان از ۲۵۵x۲۵۵

Subject :

Date .....

BREQ

(Z=1) branch if equal

C flag TST

set zero if C=1, N, ND, DF, CCR

branch if same or higher BRSR

if C=0 branch

branch if lower BRLO

branch if C=1

64 byte instruction short (branch address is not PC)

(Op) op+label 350 350 ← BRSR \* ← Branch

11101 KK KKKK K000  
+PC +1

### (Short ) Calculating Branch Address

short, BRNE, BREQ, BASH form 6 conditional branches

6 Op code 3 bits, 3 bits, 3 bits, 3 bits, 3 bits, 3 bits  
PC, Target addr, 7 bits, relative offset 2 bits  
relative offset 2 bits, relative offset 2 bits

L1-L3 L-64 in bits, relative address 11 bits

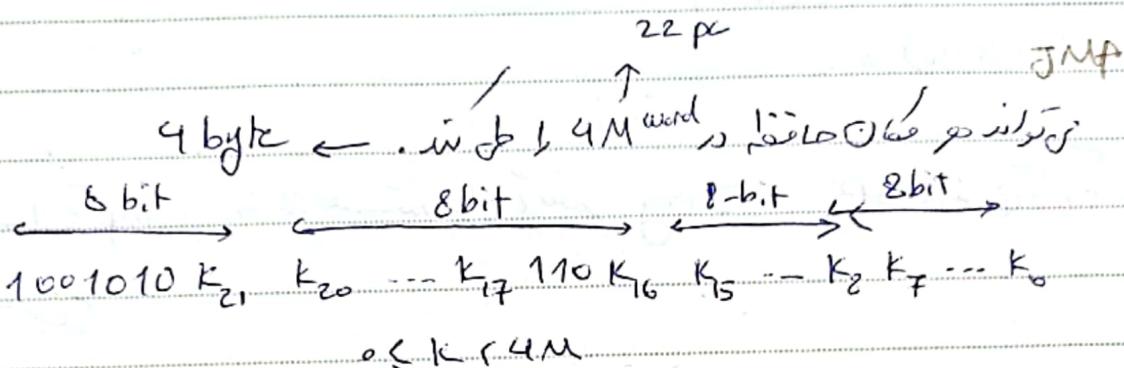
$2^{7-1} < 2^{7-1}$

MICRO\*

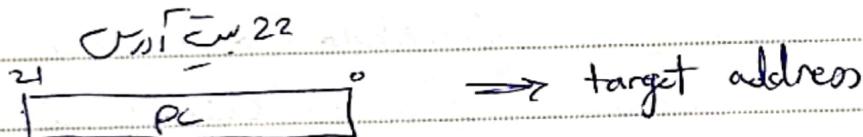
PC+1  
جاء من PC هي relative address التي ت-target address في ROM  
CPU fetch the branch CPU reads program from ROM  
in processor PC CPU finds target address

### Unconditional Branch Instruction

(indirect) IJmp, (relative) RJmp : JMP ←



Jump	31	16	\$000000 - \$3 FFFF
	15	0	



ROM address ← Rjmp

relative addr. ← lower 12 bits ← 2-byte ←

1100	KKKK	KKKK	KKKK
------	------	------	------

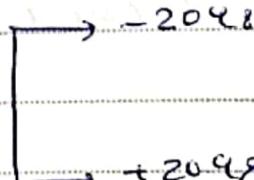
$$-2048 \leq n \leq 2048$$

$$2^{12-1} \leq n \leq 2^{12-1} - 1$$

$$-2048, 1 \quad 2047$$

$$\$000 - \$FFF$$

PC range



CPU reads PC → relative address → target address in ROM → branch to

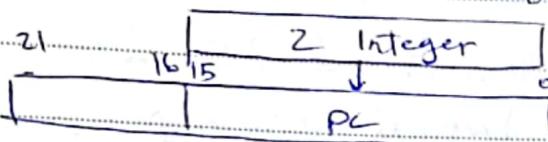
MICRO → target address in ROM → if target address is not RJMP →

(26)

indirect jmp  $\leftarrow$  IJMP

ستارچنل 6KK word  $\rightarrow$  indirect jmp IJMP منسوب زمانی PC

15



$$PC(15:0) = 2(15:0)$$

$$PC(4:16) = 0$$

أمثلة IJMP ، 2 reg هي في static ، jmp تكون نص  
مكتوب في

### Call Instructions

أمثلة subroutine ، 2 reg هي في static ، jmp تكون نص  
مكتوب في static ، المثلثات هي في static ، المثلثات هي في static

أمثلة calls

: Call

CALL (long CALL)

RECALL (relative calls)

ICALL (indirect 4 to 2)

EICALL (extended indirect call to 2)

CALL

subroutine [ ] 22 ، opcode [ ] 10  $\leftarrow$  4 byte

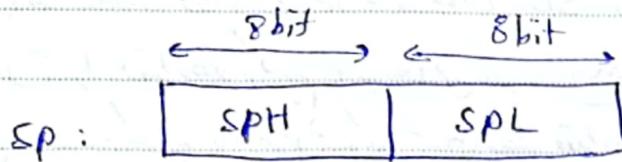
1001 010 K<sub>21</sub> K<sub>20</sub> K<sub>19</sub> K<sub>18</sub> K<sub>17</sub> 111 K<sub>8</sub> )

K<sub>15</sub> K<sub>14</sub> K<sub>13</sub> K<sub>12</sub> K<sub>11</sub> K<sub>10</sub> K<sub>9</sub> K<sub>8</sub> K<sub>7</sub> K<sub>6</sub> K<sub>5</sub> K<sub>4</sub> K<sub>3</sub> K<sub>2</sub> K<sub>1</sub> K<sub>0</sub> 05K <  $\frac{3FFFFF}{2^{22}-4M}$

MICRO 1001 010 K<sub>21</sub> K<sub>20</sub> - K<sub>17</sub> 111 K<sub>16</sub> K<sub>15</sub> - K<sub>8</sub> K<sub>7</sub> - K<sub>0</sub> 05K < 4M

in port CPU writes to RAM it goes to stack

$\text{SPH} \rightarrow \text{SPL}$  : on memory write to CPU  $\text{SP}$  is updated  
high byte low byte



$\text{SPH}, \text{SPL} \leftarrow$  nibble 256 byte is written to AVR  
 $\text{SPL}$  byte ← nibble

pushing onto the stack

$\text{push Rr}$ ,  $\text{push R10}$

only GPR (R0-R31)

$\text{SP} \rightarrow$  stack  $\downarrow$ , RR ←

pop Rr  
(LIFO)

$\text{pop Rr}$   
gpr (R0-R31)

increment → register

LDI R16, (RAMEND) → SPH → OUT SPH, R16  
HIGH

LDI R17, LOW(RAMEND) → SPL → OUT SPL, R17

## Initializing sp

لیست سریع برای معرفی داده های AVR  
 اینکه SRAM را با حافظه خارجی C تفاوت ندارد  
 درست کردن می خواهد حافظه خارجی را با RAMEND (آخرین پوزیشن حافظه) مطابقت کند.

LDI R16, HIGH(RAMEND)

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

} initializing sp

برای اجرای subroutine باید از stack استفاده کرد و call را اجرا کنیم.  
 این push را برای subroutine ایجاد کنیم.

SPL over SPH با 16 بیتی (8 bit PC) برای AVR  
 (128 > ATmega32)  $\leftarrow$  این push stack است  
 ← call 24 با 16 بیتی (8 bit PC) برای AVR  
 این push stack ۳ MSB over middle of MSB ۸ bit

RET

برای اجرای subroutine باید از stack استفاده کرد و load PC  
CALL را اجرا کرد و آنرا stack SP بود و load PC  
 باید از stack SP اجرا کرد.

Subject :

Date .....

0x55, 0xAA goes high. in toggle, PORTB goes low. with a delay in between period.

.INCLUDE "M32DEF.INC"

.ORG 0

LDI R16, HIGH(RAMEND)

OUT SPH, R16

LDI R16, LOW(RAMEND)

OUT SPL, R16

Back:

LDI R17, 0x55

OUT PORTB, R17

CALL DELAY

LDI R17, 0xAA

OUT PORTB, R17

CALL DELAY

RJMP BACK

.ORG 0x300

DELAY: LDZ R20, 0xFF

AGAIN:

NOP

NOP

DEC R20

BRNE AGAIN

RET

0x60 نحوه دعوة الى ميموري I/O Mem و GPR فيها في Stack  
call, interrupt, stack

مدى دعوة 4M word CALL serves as

address 12 bits )  $2^{12} < 2048 < 2^{20}$  address  $\leftarrow$  2-byte  $\rightarrow$  RCALL

RCALL, 4M view CALL  $\rightarrow$  EICALL, RCALL  $\rightarrow$  long jump 4K views

ICALL

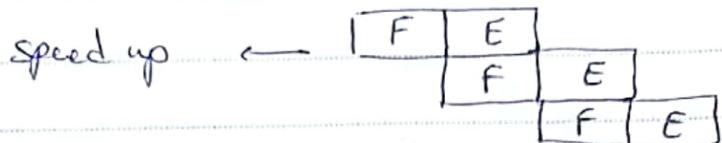
extended  $2^{22}$   $64$   $\rightarrow$  z-byte  $\rightarrow$   
injected, reading PC to 1, 2 bytes  $\rightarrow$  16 bit register  
EICALL  $\rightarrow$  forward it in program memory  $\rightarrow$  AVR  $\rightarrow$   
using EIND  $\rightarrow$  data 21-16  $\downarrow$  I/O memory

Delay for AVRs

(program) XTAL1, XTAL2  $\rightarrow$  XTAL2  $\leftarrow$  XTAL1 (1)  
 in one clock

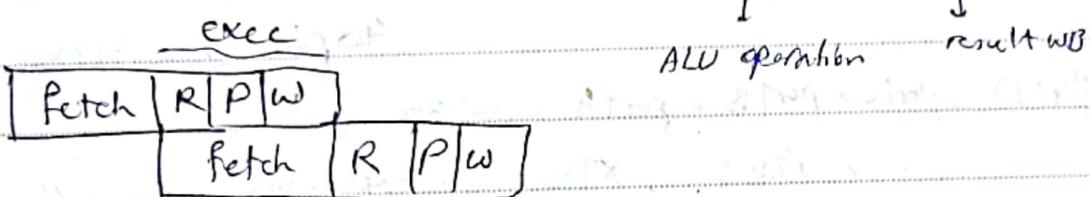
fixed  $\rightarrow$  RISC (1)      Pipeline (5) } Single instruction cycle (4)

## Pipelining



Each stage consists - اجزاء كلها

read operands + process + write back execution



## Machine cycle

machine cycle ← دورة عمل في CPU في كل دورة

(CALL, JMP) (4 to 3 µs) 1 clock cycle ← دورة

$$\text{machine cycle} = \frac{1}{\text{oscillator freq}}$$

## Branch penalty

prefetch unit in pipeline buffer with pipelining

branch unit queue exists in flush w/ CPU

queue jobs in fetch unit for code unit

for fetch unit in next execute unit discarding

branch penalty ←

in 1st clock cycle 4 to 2 stages

→ JMP, CALL, RET, on branch → BRNE, BRSL, BRLO

\* directives → 0 JMP → 3 (bytes) CALL → 4 LDS → 2  
 RJMP, IJMP → 2 BT → 1 RET → 4 RECALL, ICALL → 3 POP, PUSH → 2

: delay ایجاد

counter تعداد

loop (ر)

و clock cycle 1 branch اولیه اخیر (جی) ← ایجاد

## \* Chapter 4: AVR I/O port programming \*

40pin ← ATmega32  
PortD, PortC, PortB, PortA ← 32

X<sub>TAL1</sub>, X<sub>TAL2</sub>, GND, VCC ←

و باین یک پین شروع به این پین ها را ب AVR می بینیم و ب پورت که

و ب پورت B بیش از 8 باین AVR ←  
PortA ← F ← 2 باین 64 ← n ← 5

A-L ← 16pin ←

4port ← 40pin ←

serial communication interrupt، timer، ADC، فریکونسی، 6

I/O ← 1 (یک) output ← PORTB، DDRB  
0 (صفر) input ← PINB

و ب پورت دوستی 8 باین I/O ←

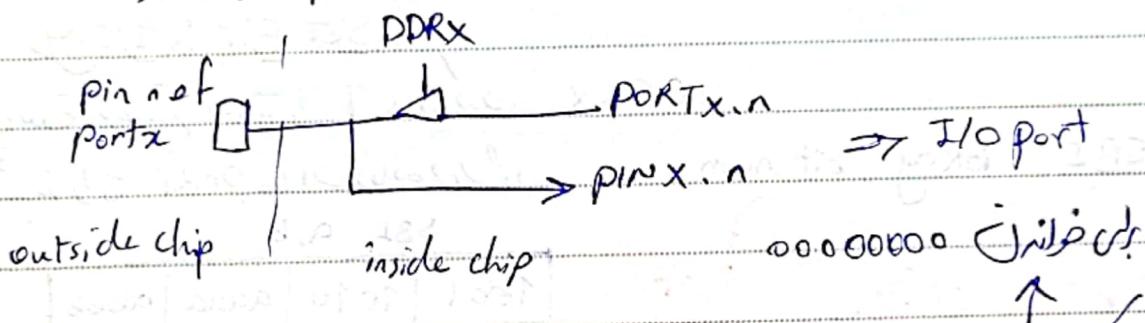
و ب پورت دوستی 8 باین I/O ←

DDRX

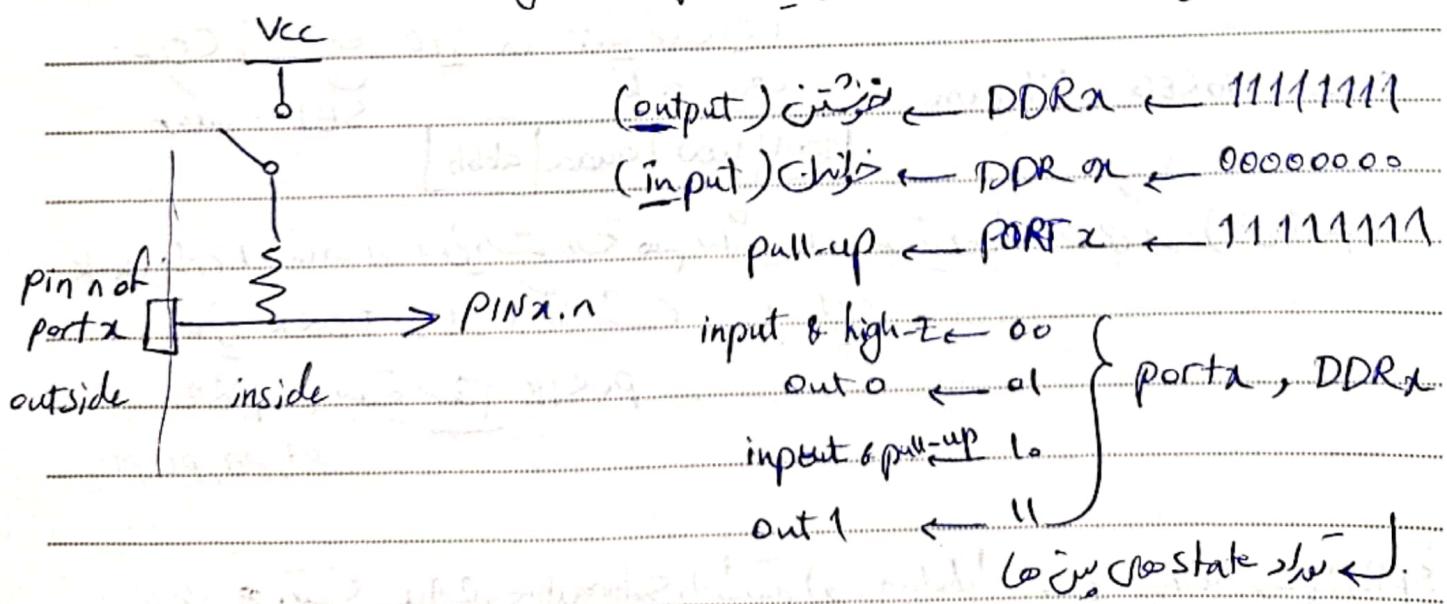
inward connection of A-D port is shown  
 In Port A, DDRX is connected to port A and its  
 output  $\leftarrow$  port B  $\leftarrow$  DPRB  $\leftarrow$  0b1111111

DDR register role in inputting

IN 0b1111111  $\rightarrow$  Port A  $\leftarrow$  DDRX (in)  $\leftarrow$  port B  $\leftarrow$  DPRB  
 In Port B, input no  $\leftarrow$  0b1111111  $\rightarrow$  TDR  $\leftarrow$  reset



in Port A, 0b1111111 is AVR pin with pull-up enable and  
 In Port B, 0b1111111 is AVR pin with pull-up enable PORTB  
 with pull-up, with enable, with high-impedance



Pull-up  $\leftarrow$  pin IN  $\leftarrow$  pin OUT  $\leftarrow$  pin I/O

in  $\leftarrow$  Pin 8, ATmega 32, 16 port no \*

### I/O ports bit-addressability

SBIS, SBIC, CBI, SBI  $\leftarrow$

ATmega32  $\leftarrow$  I/O port register

(Set Bit in I/O reg) SBI

GPRX  $\leftarrow$  set I/O port value

SBI ioReg, bit num  $\leftarrow$  PORT GPRX

$\downarrow$   $\downarrow$  SBI a,b

I/O Cleat 32

0-7

1001	1010	aaaa	abbb
------	------	------	------

- (a<sub>3</sub>b<sub>1</sub>)  $\rightarrow$  5 bit
- (b<sub>7</sub>b<sub>7</sub>)  $\rightarrow$  3 bit

SBI PORTB, 5 ; PORTB(5)=1

(Clear bit in I/O reg) CBI

CBI a,b

SBI GPRX

1001	1000	aaaa	abbb
------	------	------	------

(portB)  $\leftarrow$  8 bit value in memory location  $\leftarrow$  RAM

SBI 1  $\leftarrow$  CBI 1 after DDRB give

PORTB  $\leftarrow$  PB2

PA, PB, PC, PD

SPH, SPL, RET  $\leftarrow$  ('delay')  $\leftarrow$  Subroutine, CALL  $\leftarrow$  \* \*