به نام خدا

آشنایی با زبان اسمبلی AVR

آدرس دهی بیتی

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

آدرس دهی بیتی

- Many µP allow programs to access registers in byte size only!
  - To check a single bit
    - Read the entire byte
    - Manipulate the byte with logic instructions

- This is not the case with AVR
  - Bit-addressability options of AVR family

خیلی از میکروپروسسورها امکان دستیابی به رجیسترها رو به صورت بایتی فراهم میارن
اگر بخوایم به یک بیت خاص از یک رجیستر دستیابی دسترسی داشته باشیم مجبوریم کل اون بایت رو بخونیم و با یکسری اعمال منطقی اون بیت مورد نظر را به دست بیاریم و با اون کار بکنیم

توی خانواده های AVR این داستان می تونه با سهولت بیشتری انجام بشه
خانواده AVR امکان ادرس دهی بیتی رو برای کاربران فراهم اورده و یکسری دستوراتی در این رابطه هست که توی این اسلاید می گیم

SBR – Set Bits in Register

- Sets specified bits in register Rd.

Operation:

(i) Rd ← Rd v K

	Syntax:	Operands:	Program Counter:
(i)	SBR Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

Words	1 (2 bytes)
Cycles	1

این دستورات رو می تونیم جز دستوراتی قرار بدیم که میاد با رجیسترهای همه منظوره کار میکنه
این دستور میاد با رجیسترهای همه منظوره کار میکنه
Rd اون رجیستری است ک می خوایم مقدار بیت های اون رو تغییر بدیم
و K یک عدد 8 بیتی هستش
و Rd یک رجیستر همه منظوره است
این دستور میاد مقدار اون عدد K با مقدار رجیسترهای Rd اور یا or میکنه و توی Rd می ریزه:
اتفاقی که می افته این است که اون بیت هایی که یکه هیچ اتفاقی براش نمی افته ولی بیت هایی که صفر است: اگر بیت متناظر با اون توی K برابر یک باشد این مقدار در Rd هم یک خواهد بود و این ینی اون بیت هایی که میخوایم رو می تونیم ست بکنیم

CBR – Clear Bits in Register

- Clears the specified bits in register Rd.

Operation:

(i) Rd ← Rd • ($FF - K)

	Syntax:	Operands:	Program Counter:
(i)	CBR Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255	PC ← PC + 1

16-bit Opcode:

0111	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

Words	1 (2 bytes)
Cycles	1

این دستور روی رجیسترهای همه منظوره عملیات انجام میده
و این دستور این امکان رو به ما میده که یکسری بیت های خاصی که مدنظرمون است رو صفر بکنیم و اتفاقی که می افته اینه که:
میاد اون رجیستر مورد نظر ما رو ینی Rd با یک عدد دیگری که از K ساخته میشه and میکنه

BST – Bit Store from Bit in Register to T Flag in SREG

- Stores bit b from Rd to the T Flag in SREG (Status Register). T → Temporary

Operation:

(i) T ← Rd(b)

	Syntax:	Operands:	Program Counter:
(i)	BST Rd,b	0 ≤ d ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

16-bit Opcode:

1111	101d	dddd	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	⇔	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

توی رجیستر وضعیت یک پرچم به اسم T داریم که این T و این رجیستر T برای ذخیره سازی موقت یک بیت می تونه مورد استفاده قرار بگیره
این دستور میاد می تونه یک بیت رو از یک رجیستری که مشخص می کنیم به عنوان اپرند اون ینی Rd برداره و کپی بکنه توی رجیستر T توی اون رجیستر وضعیت
b شماره اون بیت است و b بین صفر تا 7 هستش چون رجیستر ما 8 بیتی است

توضیح دستور:
بیت شماره b از رجیستر Rd کپی میشه داخل فلگ T

BLD – Bit Load from the T Flag in SREG to a Bit in Register

- Copies the T Flag in the SREG (Status Register) to bit b in register Rd.

Operation:

(i) Rd(b) ← T

Syntax:	Operands:	Program Counter:
(i) BLD Rd,b	0 ≤ d ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

16 bit Opcode:

1111	100d	dddd	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

این دستور لود میکنه از پرچم T به یک بیت از یک رجیستر دیگری ینی Rd(b)
ینی مقدارپرچم T کپی میکنه در بیت شماره b رجیستر Rd

SBRC – Skip if Bit in Register is Cleared

- This instruction tests a single bit in a register and skips the next instruction if the bit is cleared.

(i) If Rr(b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBRC Rr,b	0 ≤ r ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip
			PC ← PC + 2, Skip a one word instruction
			PC ← PC + 3, Skip a two word instruction

Words 1 (2 bytes)

Cycles 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1111	110r	rrrr	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

این دستور میاد دستور بعدی خودش رو در صورتی که یک بیت خاصی در یک رجیستری clear باشه skip میکنه
اتفاقی که می افته اینه که:
بیت شماره b ام از رجیستر Rr رو چک میکنه و اگر برابر صفر بود از دستور بعدی پرش میکنه و یک دستور می ره جلوتر
pc ما سه حالت داره:
اگر b ما clear باشه این یک پرش انجام میده که این پرش وابسته به دستور بعدی هستش که میتونه یک خونه یا دو خونه باشه
و اگر این b ما clear نباشه دستور طبق روال عادی خودش pc رو یکی افزایش میده و می ره سراغ دستور بعدی
پس این دستور می تونه در یک سیکل ساعت یا دو سیکل ساعت یا سه سیکل ساعت اجرا بشه

SBRS – Skip if Bit in Register is Set

- This instruction tests a single bit in a register and skips the next instruction if the bit is set.

(i) If Rr(b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBRS Rr,b	0 ≤ r ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip
			PC ← PC + 2, Skip a one word instruction
			PC ← PC + 3, Skip a two word instruction

Words 1 (2 bytes)

Cycles 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1111	111r	rrrr	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

این دستور میاد:
در صورتی دستور بعدی رو skip می کنه که اون set باشه ینی برابر با یک باشه
اتفاقی که می افته:
اگر بیت b از رجیستر Rr برابر با یک باشه:
این پرش میکنه دستور بعدی رو که توی این حالت pc میتونه دو واحد زیاد بشه یا سه واحد و
اگر این بیت set نشده باشه pc ما یکی زیاد میشه و این ینی می ره سراغ دستور بعدی

SBI – Set Bit in I/O Register

- Sets a specified bit in an I/O Register. This instruction operates on the lower 32 I/O Registers addresses 0-31.

(i) I/O(A,b) ← 1

	Syntax:	Operands:	Program Counter:
(i)	SBI A,b	$0 \le A \le 31, 0 \le b \le 7$	PC ← PC + 1

16-bit Opcode:

1001	1010	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	2

این دستور از رجیسترهای I/O استفاده میکنه
این دستور میاد یک بیت رو از یک رجیستر از فضای I/O میاد یک میکنه یا set میکنه
این دستور می تونه روی 32 رجیستر پایین فضای ادرس I/O مون عمل بکنه
نحوه استفاده:
یک رو درون اون بیتی از رجیستر مورد نظر ما قرار میده

CBI – Clear Bit in I/O Register

- Clears a specified bit in an I/O register. This instruction operates on the lower 32 I/O registers addresses 0-31.

(i) I/O(A,b) ← 0

	Syntax:	Operands:	Program Counter:
(i)	CBI A,b	0 ≤ A ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1

16-bit Opcode:

1001	1000	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	2

این دستور میاد یک بیت مشخصی رو از یک رجیستری در فضای I/O پاک میکنه و صفرش میکنه
این دستور هم روی 32 رجیستر پایین فضای ادرس عمل می کنه

SBIC – Skip if Bit in I/O Register is Cleared

- This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is cleared. This instruction operates on the lower 32 I/O Registers addresses 0-31.

(i) If I/O(A,b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBIC A,b	0 ≤ A ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip
			PC ← PC + 2, Skip a one word instruction
			PC ← PC + 3, Skip a two word instruction

Words 1 (2 bytes)

Cycles 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1001	1001	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

این دستور میاد:
اگر یک بیت خاصی در یک رجیستر مورد نظر در فضای I/O صفر یا clear باشه دستور بعدیشو skip میکنه
این دستور هم روی یکی از 32 تا رجیستر پایین فضای I/O قابل اعمال هستش
اتفاقی که می افته اینجا:
اگر b صفر باشه ینی شرط ما درسته و اگر شرط درست باشه pc ما 2 یا 3 واحد زیاد میشه ولی
اگر b صفر نباشه و اون شرط صحیح نباشه pc ما یکی زیاد میشه و می ره دستور بعدی
این دستور می تونه در یک یا دو یا سه سیکل ساعت انجام بشه

SBIS – Skip if Bit in I/O Register is Set

- This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is set. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

(i) If I/O(A,b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	SBIS A,b	0 ≤ A ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip
			PC ← PC + 2, Skip a one word instruction
			PC ← PC + 3, Skip a two word instruction

Words	1 (2 bytes)
Cycles	1 if condition is false (no skip)
	2 if condition is true (skip is executed) and the instruction skipped is 1 word
	3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1001	1011	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

این دستور میاد:
اگر اون بیتی که داره چک میکنه یک باشه دستور بعدیش رو skip میکنه
این دستور می تونه روی یکی از اون 32 رجیستر پایین فضای I/O اعمال بشه

آدرس دهی بیتی

Single-Bit (Bit-Oriented) Instructions for AVR

Instruction		Function
SBI	A,b	Set Bit b in I/O register
CBI	A,b	Clear Bit b in I/O register
SBIC	A,b	Skip next instruction if Bit b in I/O register is Cleared
SBIS	A,b	Skip next instruction if Bit b in I/O register is Set
BST	Rr,b	Bit store from register Rr to T
BLD	Rd,b	Bit load from T to Rd
SBRC	Rr,b	Skip next instruction if Bit b in Register is Cleared
SBRS	Rr,b	Skip next instruction if Bit b in Register is Set
BRBS	s,k	Branch if Bit s in status register is Set
BRBC	s,k	Branch if Bit s in status register is Cleared

BRBC – Branch if Bit in SREG is Cleared

- Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is cleared. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). Parameter k is the offset from PC and is represented in two's complement form.

(i) If SREG(s) = 0 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRBC s,k	0 ≤ s ≤ 7, -64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	ksss

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false 2 if condition is true

این دستور میاد:
اینجا یک پرش شرطی نسبی داریم
میاد یک بیتی رو در رجیستر وضعیت چک میکنه و اگر صفر باشه یا clear باشه اون پرش انجام میشه و اون پرشی هم که انجام میشه نسبیه و می تونه 64 خونه جلوتر یا عقب تر باشه که این k اینو تعیین میکنه
s شماره اون بیتی در رجیستر وضعیت هستش که ما میخوایم چک بکنیم

BRBS – Branch if Bit in SREG is Set

- Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is set. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). Parameter k is the offset from PC and is represented in two's complement form.

(i) If SREG(s) = 1 then PC ← PC + k + 1, else PC ← PC + 1

	Syntax:	Operands:	Program Counter:
(i)	BRBS s,k	0 ≤ s ≤ 7, -64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	ksss

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false 2 if condition is true

این دستور میاد:
اینبار چک میکنه اگر اون بیت یک باشه یا set باشه شرط درسته و پرش صورت میگیره
ما اینجا یک پرش شرطی داریم اونم به صورت نسبی که میتونه به 64 خونه جلوتر یا عقب تر پرش بکنه

آدرس دهی بیتی

AVR Conditional Branch (Jump) Instructions

Instruction	Action	Instruction	Action
BRCS	Branch if C = 1	BRCC	Branch if C = 0
BRLO	Branch if C = 1	BRSH	Branch if C = 0
BREQ	Branch if Z = 1	BRNE	Branch if Z = 0
BRMI	Branch if N = 1	BRPL	Branch if N = 0
BRVS	Branch if V = 1	BRVC	Branch if V = 0
BRLT	Branch if S = 1	BRGE	Branch if S = 0
BRHS	Branch if H = 1	BRHC	Branch if H = 0
BRTS	Branch if T = 1	BRTC	Branch if T = 0
BRIE	Branch if I = 1	BRID	Branch if I = 0

BSET – Bit Set in SREG

- Sets a single Flag or bit in SREG.

(i) SREG(s) ← 1

	Syntax:	Operands:	Program Counter:
(i)	BSET s	0 ≤ s ≤ 7	PC ← PC + 1

16-bit Opcode:

1001	0100	0sss	1000

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔

Words	1 (2 bytes)
Cycles	1

این دستور میاد و اون بیتی که به عنوان اپرند گرفته ینی s رو یک میکنه
پس این دستور set میکنه یک بیتی رو در رجیستر وضعیت

BCLR – Bit Clear in SREG

- Clears a single Flag in SREG.

(i) SREG(s) ← 0

	Syntax:	Operands:	Program Counter:
(i)	BCLR s	0 ≤ s ≤ 7	PC ← PC + 1

16-bit Opcode:

1001	0100	1sss	1000

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔

Words	1 (2 bytes)
Cycles	1

این دستور میاد یک بیتی رو در رجیستر وضعیت پاک میکنه

آدرس دهی بیتی

Manipulating the Flags of the Status Register

Instruction	Action		Instruction	Action	
SEC	Set Carry	C = 1	CLC	Clear Carry	C = 0
SEZ	Set Zero	Z = 1	CLZ	Clear Zero	Z = 0
SEN	Set Negative	N = 1	CLN	Clear Negative	N = 0
SEV	Set overflow	V = 1	CLV	Clear overflow	V = 0
SES	Set Sign	S = 1	CLS	Clear Sign	S = 0
SEH	Set Half carry	H = 1	CLH	Clear Half carry	H = 0
SET	Set Temporary	T = 1	CLT	Clear Temporary	T = 0
SEI	Set Interrupt	I = 1	CLI	Clear Interrupt	I = 0

آدرس دهی بیتی

- The internal RAM is not bit-addressable
- In order to manipulate a bit of internal RAM location
  - Bring it into GPR and manipulate

Write a program to see if the internal RAM location $195 contains an even value. If so, send it to Port B. If not, make it even and then send it to Port B.

```
.EQU   MYREG = 0x195          ;set aside loc 0x195
       LDI   R16,0xFF
       OUT   DDRB,R16         ;make Port B an output port
AGAIN:LDS    R16,MYREG
       SBRS  R16,0            ;bit test D0, skip if set
       RJMP  OVER             ;it must be LOW
       CBR   R16,0b00000001   ;clear bit D0 = 0
OVER: OUT    PORTB,R16        ;copy it to Port B
       JMP   AGAIN            ;we can use RJMP too
```

ایا این امکان در مورد فضای RAM داخلی هم وجود دارد یا ندارد؟
ما نمی تونیم برای خانه های رم یا RAM داخلی میکرو به صورت بیتی کار بکنیم
اما برای اینکه بخوایم و بتونیم به صورت بیتی با یک محلی توی حافظه رم کار بکنیم ما باید اونو بیاریم توی یک رجیستر همه منظوره و بعد از همین دستوراتی که توی این اسلاید گفتیم استفاده بکنیم و اون هارو تغییر بدیم یا ...
اتفاقی که داره توی کد هم می افته الان همین است

ماکروها

- A group of instructions performs a task
  - Used repeatedly
- Does not make sense to rewrite this code every it is needed

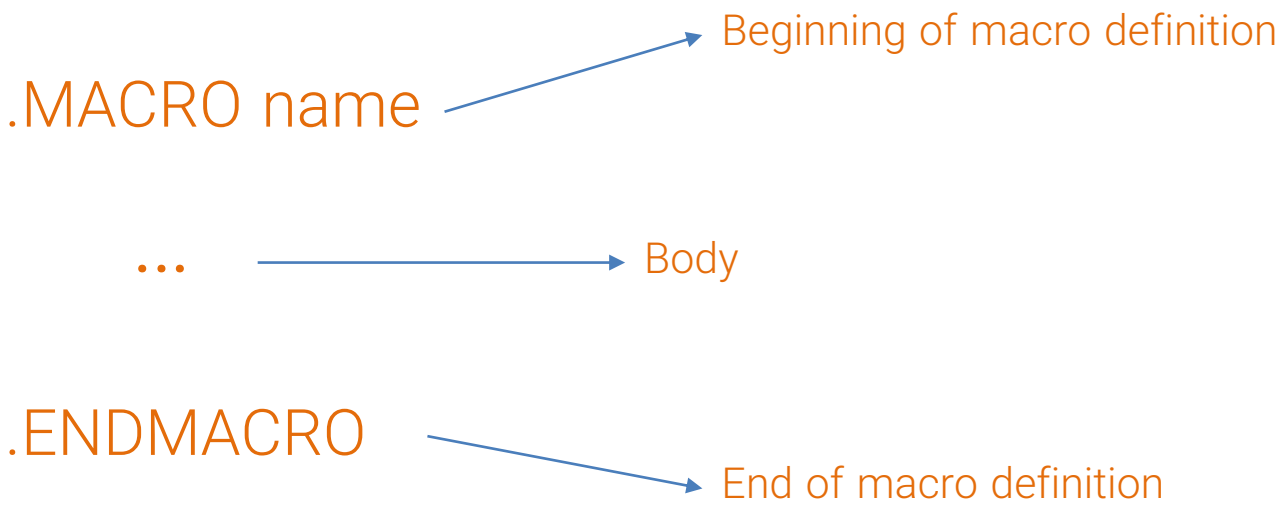
Macros

- Macros allow
  - To write the task once only, and to invoke it whenever it is needed
  - Reduce the time to write code and possibility of errors

ماکرو:
وقتی که کد می زنیم خیلی پیش میاد که اون برنامه ای که داریم می نویسیم قسمت های تکراری داره پس می تونیم از ماکروها بریم
ماکروها ابزاری هستند که به ما اجازه میدن بدون اینکه یک کد رو مرتب تکرار بکنیم اون ها رو در قالب یک ماکرو تعریف بکنیم و اون هارو صدا بزنیم ینی ماکروها رو صدا بزنیم

تعریف ماکروها

- Macro definition



- A macro can take up to 10 parameters
  - Parameters can be referred to as @0 to @9
- After the macro has been written, it can be invoked by its name

تعریف:
یک شروعی برای ماکرو داریم که با .MACRO مشخص میکنیم و بعد از اون یک نام برای اون ماکرو قرار میدیم
و بعد بدنه ماکرو هستش
و بعد از ان ماکرو رو با .ENDMACROS تموم میشه
هر ماکرو می تونه تا 10 پارامتر رو به عنوان ارگومان داشته باشه
و برای اینکه بخوایم با این ارگومان ها کار بکنیم می تونیم به صورت @0 کار بکنیم که از @0 داریم تا @9
و بعد ازاینکه یک ماکرو رو نوشتیم و برای اینکه بتونیم استفاده بکنیم کافیه که از نام ماکرو استفاده بکنیم و بعد از نام اون بیایم اونو فراخوانی بکنیم

ماکروها نحوه استفاده

For example, moving immediate data into I/O register data RAM is a widely used service, but there is no instruction for that. We can use a macro to do the job as shown in the following code:

```
.MACRO      LOADIO
        LDI   R20,@1
        OUT   @0,R20
.ENDMACRO
```

The following are three examples of how to use the above macro:

```
1. LOADIO   PORTA, 0x20       ;send value 0x20 to PORTA

2. .EQU     VAL_1 = 0xFF
   LOADIO   DDRC, VAL_1
3. LOADIO   SPL, 0x55         ;send value $55 to SPL
```

مثال:

برای اینکه این ماکرو رو فراخوانی بکنیم نام اون رو ذکر میکنیم و بعد ارگومان های مورد نیاز اونو بهش میدیم که اینجا ارگومان هاش PORTA , 0x20 است
الان 1@ ما میشه 0x20 و 0@ ما میشه PORTA

ماکروها نحوه استفاده

Assume that several macros are used in every program. Must they be rewritten every time? The answer is no, if the concept of the .INCLUDE directive is known. The .INCLUDE directive allows a programmer to write macros and save them in a file, and later bring them into any program file. For example, assume that the following widely used macros were written and then saved under the filename "MYMACRO1.MAC".

```
 toggling Port B using macros
.INCLUDE "M32DEF.INC"
.INCLUDE "MYMACRO1.MAC" ;get macros from macro file
;-----------------------------program starts
      .ORG  0
      LOADIO   DDRB,0xFF
L1:   LOADIO   PORTB,0x55
      DELAY    R18,0x70
      LOADIO   PORTB,0xAA
      DELAY    R18,0x70
      RJMP     L1
```

ممکنه که ماکروهای متعددی داشته باشیم
برای اینکه بتونیم بین برنامه های متفاوتی که داریم باز از این کدها بتونیم به صورت مشترک استفاده بکنیم می تونیم اون ماکروها رو در قالب یک فایل مجزایی ذخیره بکنیم و بعد با استفاده از دایرکتیو include اون هارو به فایل برنامه اضافه بکنیم بدون اینکه بخوایم دوباره اون کدهارو بنویسیم می تونیم با این کار اونارو به فایل برنامه اضافه بکنیم

پایان

موفق و پیروز باشید