Subject:
Year. 90 Month. 12 Date. 10 ( )

اصول میکروکامپیوتر

سرفصل ها در کلاس:

1- اصول کارکرد یک پردازنده

2- بررسی میکروکنترلرهای AVR (MEGA32)

3- برنامه نویسی به زبان C

4- بررسی و استفاده از امکانات AVR

- پورت های I/O
- اتصال LCD
- تایمر/کانتر
- ارتباط سریال
- ADC (A/D)

5- برنامه ریزی میکروکنترلرهای AVR به زبان Basic

مراجع:

① The AVR MicroControllers And Embeded Systems Using Assembly And C
M. Reza Mazidi, Printcehall 2010

② مرجع کامل میکروکنترلرهای AVR، پرتوی فر- مظاهری- پیانلو، انتشارات آتش 1388

③ میکروکنترلرهای AVR به زبان C، حمیدیادامی نژاد، انتشارات ادیستان

ارزیابی: پایانترم 60٪ میانترم 30٪ تمارین 10٪

Subject:
Year. Month. Date. ( )

فصل اول:

- سیستم کنترلی مبتنی بر پردازنده دارای سه قسمت اصلی زیر است:

(1) ورودی: دریافت داده از محیط خارج
(2) پردازنده: پردازش داده های ورودی
(3) خروجی: تولید خروجی مطلوب

- مزیت پردازنده های مجتمع بر مدارات منطقی دیجیتالی:

1- سادگی: حجم کم تر و سهولت در استفاده از آنها
2- قابلیت برنامه ریزی: تمام دستورات ورودی و خروجی و پردازشی در قالب دستورات نرم افزاری (برنامه) به پردازنده داده می شود.

- اجزای پردازنده:

1- واحد پردازش مرکزی CPU
2- حافظه
3- ورودی - خروجی

- برنامه ذخیره شده در حافظه توسط واحد CPU خط به خط اجرا می شود و خروجی مطلوب را ایجاد می کند.

- اتصال قسمت های مختلف یک پردازنده توسط Bus ها انجام می شود. (Data Bus - Address Bus)

Subject:
Year. Month. Date. ( )

- عملکرد پردازنده :

پردازش اطلاعات ذخیره شده در حافظه به صورت زیر انجام می شود :

1. خواندن دستورات از حافظه (واکشی - Fetch)
2. دکد کردن دستورات (Decode)
3. اجرا کردن دستورات (Execute)

- زبان پردازنده :

زبان قابل درک برای یک پردازنده، زبان صفر و یک (دودویی یا زبان ماشین) است. بنابراین کوچک ترین واحد اطلاعات در این زبان یک بیت (صفر یا یک) است و اطلاعات به صورت رشته ای از بیت های صفر و یک است.

مثلاً برای خواندن دو عدد از ورودی و محاسبه مجموع (با زبان صفر و یک):

خواندن یک عدد : 00110101
خواندن یک عدد : 00110101
جمع کردن : 10010001
خروجی : 01000000

- به دلیل دشوار بودن این زبان، زبان اسمبلی معرفی شد.

- زبان اسمبلی : برای هر دستور معادل انگلیسی در نظر گرفته شده است :

- In ...
- In ...
- ADD ...
- Out ...

Subject:
Year. Month. Date. ( )

- زبان اسمبلی، زبان وابسته به ماشین است. یعنی کدهای نوشته شده برای یک عملیات خاص در پردازنده های مختلف با متفاوت است. برای مثال دستور پاک کردن صفحه نمایش در زبان Basic، cls و یا در زبان C، دستور clrscr است اما در زبان اسمبلی به صورت زیر است:

```
MOV AH, 25
MOV AL, --
:
:
MOV DX, Ø
```

- برنامه ای که به زبان اسمبلی نوشته شده، توسط نرم افزاری که اسمبلر نامیده می شود به زبان صفر و یک تبدیل می شود. اگر برنامه به زبان C یا Basic باشد به این نرم افزار کامپایلر گویند.

- برنامه نویسی به زبانهای سطح بالا مثل C و Basic ساده تر است اما برنامه های نوشته شده به زبان اسمبلی حجم کم تری دارند و سرعت بالاتری را فراهم می کنند.

Subject: جلسه دوم
Year. Month. Date. ( )

اجزاء پردازنده: (ALU, CU, Registers)

اجزاء پردازنده: Register, ALU, CU

1. ثبات (Register): حافظه کوچکی هستند که در داخل پردازنده قرار دارد. تعداد رجیسترها و چندبیتی بودن آنها کارایی پردازنده را مشخص می‌کند.

الف) رجیستر انباره (Accumullator): رجیستر هم منظوره است که برای ذخیره سازی عملیات ریاضی و منطقی استفاده می‌شود.

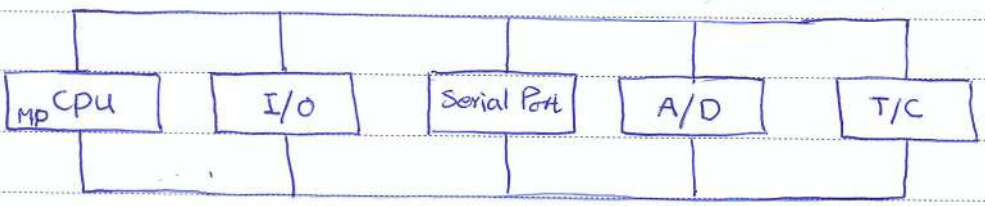
ب) رجیستر پرچم (Flag Reg.): بیت های رجیستر تغییر وضعیت در حین پردازش را نشان می‌دهد. مانند پرچم (ZF) اگر نتیجه محاسباتی صفر شود این پرچم تغییر وضعیت داده و مقدار آن یک می‌شود (ZF=1)
مثال های دیگری که می‌توان برای بیت های پرچم استفاده کرد: ZF, OVF, Sign Flag.

ج) رجیستر شمارنده: Instruction Pointer Or Program Counter (IP) or (PC)
رجیستری است که برای شمارش خطوط برنامه استفاده می‌شود، با توجه به اینکه هر برنامه خط به خط اجرا می‌شود، مقدار رجیستر شمارنده با رسیدن به هر دستور یک واحد اضافه می‌شود.

4. ALU: عملیات منطقی و ریاضی در این واحد انجام می‌شود.

3. CU: کنترل کننده سایر واحدها است و عملیات decode کردن دستورات در این واحد انجام می‌شود.

میکروپروسسور:



معایب استفاده از میکروپروسسور: ① افزایش حجم
② افزایش توان مصرفی
③ افزایش هزینه

Subject:
Year. Month. Date. ( )

میکروکنترلر:

تقطع یا چیپی است که امکانات جانبی مانند حافظه‌ها و تایمرکانترها و... را به همراه پردازنده در خود جای داده است.

انواع میکروکنترلرهای AVR:

جزء اولین خانواده‌های AVR : AT Tiny
: AT 90S
: AT Mega
: XMega

8 بیتی
16 بیتی

- به علت اینکه معماری استفاده شده در AVR از نوع معماری RISC می‌باشد، سرعت پردازش آن نسبت به سایر میکروکنترلرها بالاتر است. معیار اندازه‌گیری سرعت در این میکرو، MIPS می‌باشد که میلیون‌ها دستورالعمل را در یک ثانیه انجام می‌دهد. (هر دستور در یک پالس ساعت انجام می‌شود.)

RISC : Reduce Instruction Set Computer
MIPS : Million " Per Second

- دستورات استفاده شده در مدل‌های پایین‌تر با کمی تغییر در مدل‌های بالاتر هم قابل اجرا می‌باشد.
- میکروکنترلرهای AVR از لحاظ ولتاژ کاری در دو نوع L و معمولی موجود هستند که در مدل L ولتاژ کاری در محدوده 2.7V-5.5V و در مدل معمولی محدوده ولتاژ 4.5V-5.5V است.

2.7-5.5

Subject:
Year. Month. Date. ( )

میکروکنترلر ATMEGA32 :

امکانات میکروکنترلر ATMEGA32 :

1- 4 پورت ورودی و خروجی : پورت ها رابط دنیای بیرون و میکرو باشند که می تواند خروجی یا ورودی باشد.

2- تایمر و کانتر : T/C0 و T/C1 که 8 بیتی هستند و T/C2 ، 16 بیتی می باشد. (8 bit → 0 و 1) (16 bit → 2)

نکته: منابع تولید کلاک در میکرو ، اسیلاتور داخلی است که مقادیر 1، 2، 4، 8 MHz را دارا است و برای سرعت های بالاتر از کریستال خارجی 16 MHz در پایه 12 و 13 استفاده می شود. سرعت پردازش متناسب با کریستال (فرکانس کاری) است.

3- ارتباط سریال : (RXD , TXD)

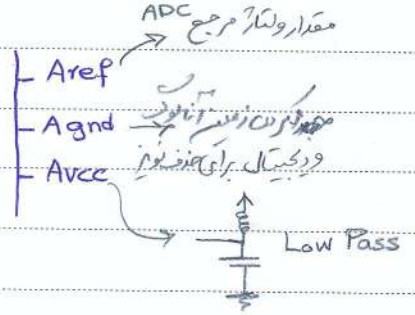
4- منابع وقفه : در هنگام وقوع وقفه، میکرو عملیات را متوقف و زیر برنامه مربوط به وقفه را اجرا می کند.
هر یک امکانات میکرو وقفه مربوط به خود را دارد مانند وقفه T/C و A/D و ...

5- مولد موج PWM : (OCR1a, OCR1b)

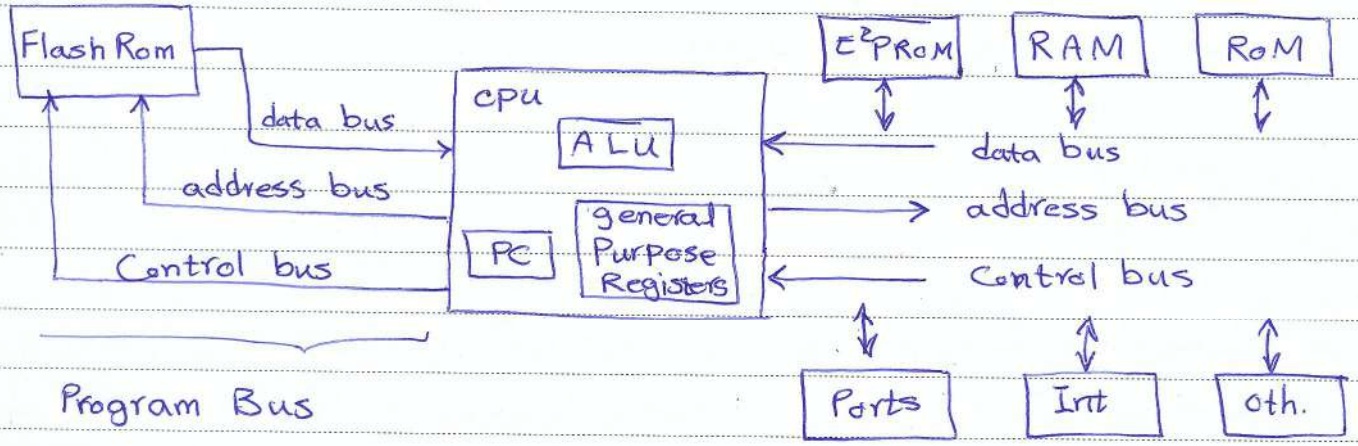
Pulse Width Modulation

Subject:
Year. 1391 Month. 1 Date. 15 ( )

6- مبدل آنالوگ به دیجیتال: A/D کانال 8



فصل سوم: معماری میکروکنترلر



انتقال و پردازش اطلاعات در هر پردازنده از طریق ارتباط گذرگاهها با cpu انجام می‌شود، انواع گذرگاههایی که در هر پردازنده وجود دارند عبارتند از:

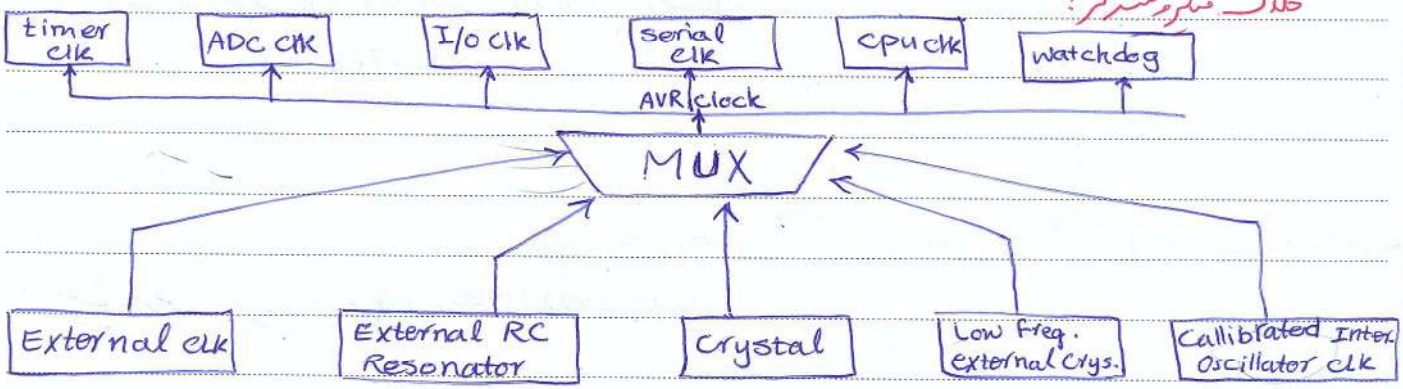
1- گذرگاه داده: تمامی اطلاعاتی که در سیستم پردازش می‌شود از باس داده عبور می‌کند. (برقرارکننده ارتباط میان پردازنده و حافظه)

2- گذرگاه آدرس: مشخص کننده این است که اطلاعات در اختیار چه قسمتی قرار می‌گیرد.

3- گذرگاه کنترل: کنترل کننده ارتباط بین باس داده و آدرس است.

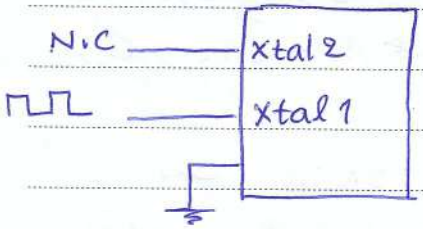
Subject:
Year. Month. Date. ( )

کلاک میکروکنترلر:

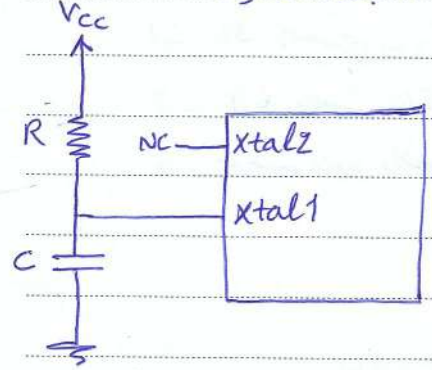


کلاک میکرو از منابع زیر تولید می‌شود که با انتخاب هر منبع و فیوزبیت مربوط به آن می‌توان میکرو را راه‌اندازی کرد:

۱- منبع کلاک خارجی: با دادن پالس به پایه XTAL1 و بازبودن XTAL2 می‌توان از کلاک خارجی به عنوان کلاک میکرو استفاده کنیم.

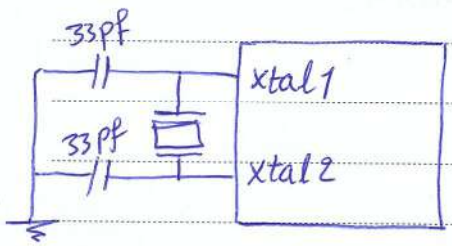


۲- اسیلاتور RC خارجی: با اتصال مدار زیر و رابطه $f \cong \frac{1}{3RC}$ بدست می‌آید.



نکته: $Min(C) \cong 22pf$

۳- کریستال خارجی: با استفاده از مدار روبرو و کریستال با فرکانس 16MHz بدست می‌آید.

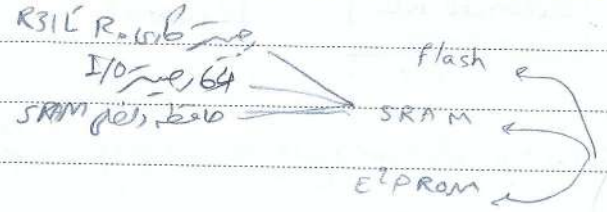


Subject:
Year. Month. Date. ( )

۴- کریستال خارجی فرکانس پایین:
مانند مدار کریستال خارجی است با فرکانس کریستال
32.768 KHz

۵- اسیلاتور کالیبره شده داخلی: دارای فرکانس های 8^{MHz} و 4 و 2 و 1^{MHz} می باشد، که به صورت پیش فرض میکرو روی فرکانس 1MHz داخلی قرار دارد.

حافظه میکروکنترلر:

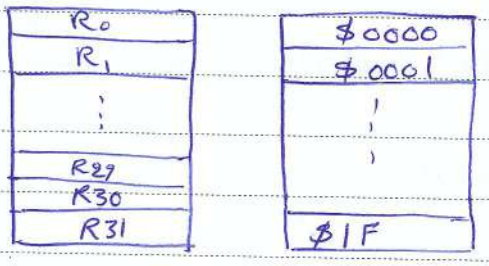


۱- Flash Rom (Program Memory):
حافظه ای است که برنامه در آن ذخیره می شود. با پروگرام کردن میکرو، حافظه فلش پر می شود و خارج کردن میکرو از مدار پروگرامر، امکان تغییر حافظه وجود ندارد. این حافظه با قطع برق از بین نمی رود و حدود 10000 بار قابلیت نوشتن و پاک کردن دارد.
در میکرو ATMEGA16 حجم این حافظه 16KB می باشد.

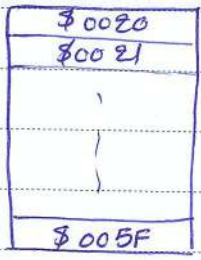
۲- حافظه SRAM: فضایی است که میکرو در هنگام اجرا کردن برنامه، متغیرها را در این فضا ذخیره می کند. با قطع برق اطلاعات این حافظه پاک می شود. اطلاعات توسط CPU در این حافظه نوشته می شود و یا از آن خوانده می شود. حافظه SRAM به سه بخش تقسیم می شود:

۱- رجیستر کاری R0 تا R31 32
۲- رجیستر I/O 64
۳- حافظه داده داخلی SRAM

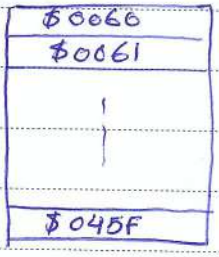
Register Files
(name) Data Address Reg.



I/O Register



Internal SRAM



Subject:
Year. 91 Month. 1 Date. 22 ( )

Mega16 512 byte Mega32 1024 byte

3- حافظه E^2PROM:

برای ذخیره سازی داده هایی که می خواهیم با قطع برق از بین نرود، می توان از این نوع حافظه استفاده کرد. با استفاده از دستورات لازم داده ها را در این حافظه می نویسیم. در Mega16 حجم این حافظه 512 byte می باشد. این حافظه 100 هزار بار قابلیت پاک شدن دارد.

منابع Reset:

میکرو برنامه در حال اجرا را متوقف کرده و از آدرس Reset شروع به کار می کند.

1- Reset شدن میکرو هنگام روشن کردن

2- ریست خارجی، وصل کردن پایه 9 به زمین.

3- Watchdog Reset: برای جلوگیری از هنگ کردن میکرو از کانتر Watch dog استفاده می شود. این کانتر با شمارش و هنگام رسیدن به مقدار Max، میکرو را ریست می کند.

4- Burn_out Reset: اگر ولتاژ میکرو از حد آستانه کمتر شود، میکرو Reset می شود.
Brownout

Subject:
Year. Month. Date. ( )

زبان C :

زبان C، زبانی با سطح میانی است که برای برنامه ریزی میکروکنترلرهای AVR قابل استفاده است. مزیت های این زبان بر زبان اسمبلی عبارتند از :

1- سادگی زبان C

2- انعطاف پذیری : یعنی با نوشتن برنامه به زبان C، نیازی به دانستن ساختار داخلی میکروپنت و با تغییر (ی) جزیی می توان برنامه نوشته شده برای میکروکنترلر خاص را برای میکروهای دیگر نیز استفاده کرد.

3- دانستن دستورات کمتر

4- دسترسی به کتابخانه ها و توابع موجود

5- قابل فهم تر بودن برنامه زبان C

- مزیت استفاده از زبان اسمبلی بر C این است که در یک برنامه خاص، حجم کدهای تولید شده توسط اسمبلر، نسبت به کدهای تولید شده توسط کامپایلر حداقل 1/5 تا 2 برابر کمتر است. لذا اگر هدف، سرعت و حجم کد پایین تر باشد بایستی از زبان اسمبلی استفاده کنیم.

نکات در مورد زبان C :

1- به جز دستوراتی که با علامت # شروع می شوند، تمامی دستورات با ; خاتمه می یابند.

```
a = a + b ;
```

2- برای نوشتن عبارت توضیحی :

```
// توضیحات
/* توضیحات */
```

3- استفاده از آکولاد : هر آکولاد که باز می شود باید در جایی بسته شود.

```
if a == 8 {
   ===
}
```

Subject:
Year. Month. Date. ( )

توابع کتابخانه Header ← سرآیند

4- در یک خط می‌توان چندین دستور را نوشت:

```
a=a+b , a=a-b ;
```

5- تعریف متغیر قبل از استفاده متغیر است.

6- برای نامگذاری متغیرها می‌توان از حروف بزرگ و کوچک، اعداد و علامت Underline هم استفاده کرد. طول هر متغیر 32 کاراکتر است.

7- برای تعریف متغیرهای عددی از 3 شکل زیر استفاده می‌شود:

```
x = 255 ; // decimal
x = 0B11111111 ; // Binary
x = 0xFF ; // HexaDecimal
```

ساختار کلی برنامه:

- شروع هر برنامه با فایل‌های سرآیند (Header Files) می‌باشد، این فایل‌ها اطلاعاتی را به ابتدای برنامه الحاق می‌کنند. در واقع فایل‌های سرآیند توابعی از پیش تعریف شده می‌باشند.

```
# include <mega16.h>
```

- ساختار یک برنامه زبان C:

```
# include <فایل‌های سرآیند.h>
# include <2~ ~.h>
Global Var;
Defining Func;
int main Void {
Lacal Var;
Instructions;
}
```

- بعد از تعاریف کلی، main برنامه را داخل حلقه while می‌نویسیم. مقداردهی اولیه به متغیرها و رجیسترها قبل از حلقه انجام می‌شود.

```
while {
}
```

Subject:
Year. Month. Date. ( )

انواع و محدوده داده ها:

- به دلیل اینکه عملیات میکرو به صورت 8 بیتی انجام می شود، بعد متغیرها به صورت 8 بیتی یا ضریبی از 8 تعریف می شود.

نوع متغیر	تعداد بیت	محدوده
bit	1	0, 1
(signed) char	8	-128 to 127
unsigned char	8	0 to 255
signed int	16	-32768 to 32767
unsigned int	16	0 to 65535
(signed) Long	32	-2147483648 to 2147483647
unsigned Long	32	0 to 4294967295
float	32	

انواع متغیرها:

۱- متغیرهای سراسری: قابل دسترس در تمام توابع برنامه
۲- به محلی: فقط در توابعی که تعریف می شوند، قابل استفاده است.

کلاس:

۱- متغیرهای Static: متغیرهایی هستند که در فراخوان های مختلف توابع، مقدار ثابت دارد.

```
static int n=1;
```

۲- متغیرهای Extern: با تعریف این متغیر، حافظه ای تشکیل می شود که قابل دسترسی است.

```
extern int a;
```

SRAM

3- متغیر eeprom: اگر هدف تعریف متغیرهایی باشد که با قطع برق از بین نرود، می توان متغیر را از نوع eeprom تعریف کنیم.

```
eeprom نوع متغیر ; نام متغیر
```

```
eeprom int a;
```

Subject:
Year. 91 Month. 1 Date. 23 ( )

4- متغیرهای فلش (Flash): متغیرهایی هستند که فقط قابل خواندن هستند.

```
Flash int a;
Const
```

حافظه
تعیین آدرس ذخیره داده در SRAM:

```
int y @0x80;
```

ذخیره متغیر y در آدرس 80H

آرایه ها: به یک سری از متغیرهای هم نوع آرایه می گویند. آرایه را می توان در فضای RAM یا E^2PROM ذخیره کرد.

- فرمت معرفی آرایه یک بعدی:

نوع آرایه | اسم آرایه | [طول آرایه]

```
int s[3]   OR   int s[]=[1,4,7,3]
```

- برای معرفی آرایه دو بعدی:

نوع آرایه | اسم آرایه | [بعد اول] | [بعد دوم]

```
int s[4][3]
```

- نحوه ذخیره آرایه دو بعدی به صورت سطری است.

عملگرها:

1- عملگرهای محاسباتی:

3	+	a+b	
	-	a-b	
2	x	a*b	int x,y; x=10; y=++x; → y=11, x=11
	/	a/b	int x,y; x=10; y=x++; → y=10, x=11
	%	a%b	
1	++	a++	
	--	a--	

Subject:
Year. Month. Date. ( )

2- عملگرهای رابطه ای : مرتبط کننده دو عملوند.

$>$ $x > y$

$>=$ $x >= y$

$==$ $x == y$

$!=$ $x != y$

3- عملگرهای منطقی:

!	not	!x
&&	and	x && y
||	OR	x || y

تقدم:
1. !
2. $>$ $>=$
3. ==, !=
4. &&
5. ||

4- عملگرهای بیتی:

and	&
OR	|
not	~
XOR	^
Right shift	>>
Left	<<

تعداد شیفت >> اسم متغیر

a= 0b 0000 0001

y = a>>1 y = 0b 1000 0000

دستورات پیش پردازنده:

1- دستور #inclue : برای ضمیمه کردن فایل های سرآیند از دستور (include) استفاده می شود که این فایل ها به دو صورت وجود دارد:

الف) فایل هایی که در کامپایلر موجود است: # include< mega8.h>

ب) فایل هایی که توسط برنامه نویس اضافه می شود: # include "Num.h"

Subject:
Year. Month. Date. ( )

2- define : این دستور به صورت سمبلیک عبارتی را با عبارت دیگر جایگزین می کند.

define LED Portd.0 ; # define LED Portd.0

کلمه LED را با Portd.0 جایگزین می کند.

undef LED

define LED Portd.0
undef LED

- حلقه های تکرار:

۱- حلقه تکرار for:

for (مقدار اولیه شمارنده ; شرط حلقه ; گام شمارش) {
دستورات
}

حلقه بی نهایت:

```
for( ; ; ){
}
```

2- حلقه while:

while (شرط) {
}

ابتدا شرط حلقه چک می شود، در صورتی که شرط درست باشد، دستورات اجرا می شود.

حلقه بی نهایت:

```
while (1){
}
```

3- حلقه Do-while:

Do {
} while (شرط)

ابتدا دستورات اجرا می شود بعد شرط حلقه چک می شود.

دستورات شرطی:

۱- If:

If (شرط) {
}

2- If-else

If (شرط) {
دستورات ۱ ;
}
else if (شرط) {
دستورات ۲ ;
}
else {
دستورات ۳ ;
}

Subject:
Year. Month. Date. ( )

Switch - Case - 3:

```
switch ( عبارت ) {
    Case مقدار1 :
    <دستورات 1>
     break;

    Case مقدار2 :
    <دستورات 2>
     break;
     :
    default {
    <دستورات N>
    }
```

۱- با دستور break می‌توان از حلقه خارج شد.
۲- نوشتن default اختیاری است.
۳- مقادیر Case ها باید متفاوت باشند.
۴- از دستور switch می‌توان به صورت تودرتو استفاده کرد.

نوشتن برچسب:

```
{ Lable_Name :
  ____
  ____
  goto xy;
       LableName
```

با استفاده از دستورات goto می‌توان به برچسب پرش کرد.

goto اسم برچسب

مثال:

```
while(1){
    if a==10  goto xy;
    a++;
    }
xy:
   Portd.0=1
```

Subject: سه‌شنبه
Year. 139 | Month. 1 Date. 29 ( )

جلسه ششم

نحوه نوشتن توابع:

فرمت نوشتن یک تابع به صورت زیر است:

نوع تابع نام تابع (پارامترها) {

و دستورات

}

- هر تابع براساس هدفی که نوشته می‌شود دارای ورودی و خروجی است. به ورودی‌های تابع آرگومان تابع می‌گویند و به مقداری که تابع برمی‌گرداند، مقدار برگشتی آن تابع گویند.

1- هر تابع قبل از main نوشته می‌شود.
2- معرفی ورودی‌های تابع در تابع صورت می‌گیرد.
3- برای فراخوانی تابع، از نام تابع استفاده می‌شود.
4- در صورتی که تابع ورودی نداشت از Void استفاده می‌کنیم.

- توابع فراخوانی شده در برنامه براساس مقدار بازگشتی به دو دسته تقسیم می‌شوند:

الف) توابعی که یک مقدار بازگشتی دارند. برای بدست آوردن مقدار بازگشتی از دستور Return استفاده می‌شود.

عبارت Return ;

ب) توابعی که مقدار بازگشتی ندارند، از نوع Void تعریف می‌گویند

مثال: تابعی بنویسید که $w=\frac{x*y}{f}$ را برحسب 5 و 12 و 1 محاسبه کند.

```
#include <Mega32.h>
float g,w;
float name (int x, int y, int f){
g= x*y;
g=g/f;
return (g);
}
```

```
Void main (Void){
while (1){
w = name (1, 12, 5);
}
}
```