



# ریزپردازنده و زبان اسمبلی

---

دانشکده کامپیوتر دانشگاه یزد

نیم‌سال دوم تحصیلی ۹۶-۹۷

ارائه‌دهنده : پریسا استواری



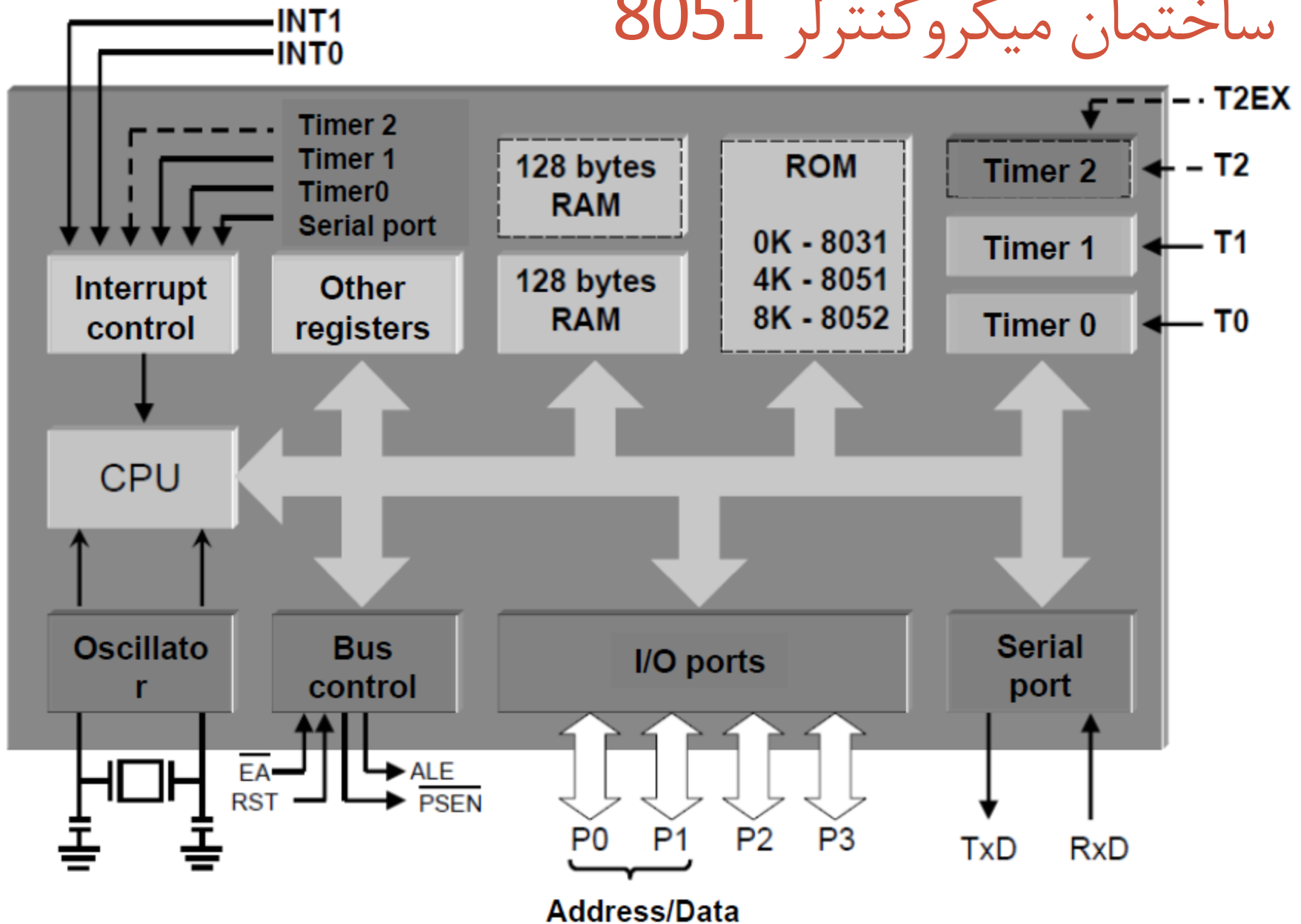
# سخت افزار میکروکنترلر 8051

---

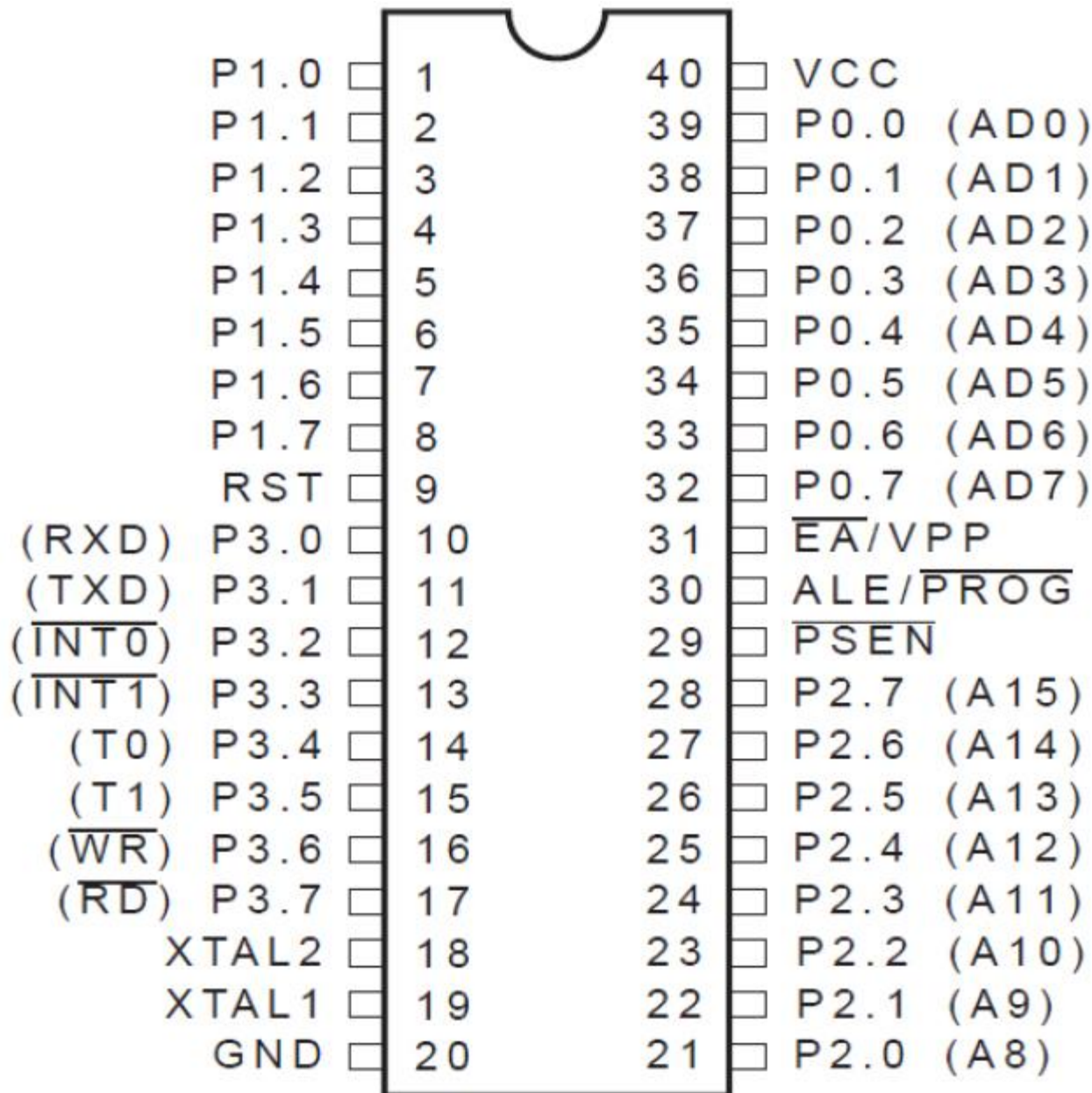
## میکروکنترلر 8051

- 4k بایت حافظه ROM داخل تراشه (برای قرارگیری کد برنامه)
- 128 بایت حافظه RAM داخل تراشه (برای داده‌های برنامه)
- چهار پورت ورودی خروجی (۸ بیتی)
- دو تایمر ۱۶ بیتی
- پورت سری
- 64k بایت فضای حافظه برنامه خارجی
- 64k بایت فضای حافظه داده خارجی
- عملیات بر روی بیت‌ها
- ۲۱۰ بیت آدرس‌پذیر در حافظه
- انجام عملیات ضرب و تقسیم در ۴ میکرو ثانیه

# ساختمان میکروکنترلر 8051



## 8051 Pin Diagram



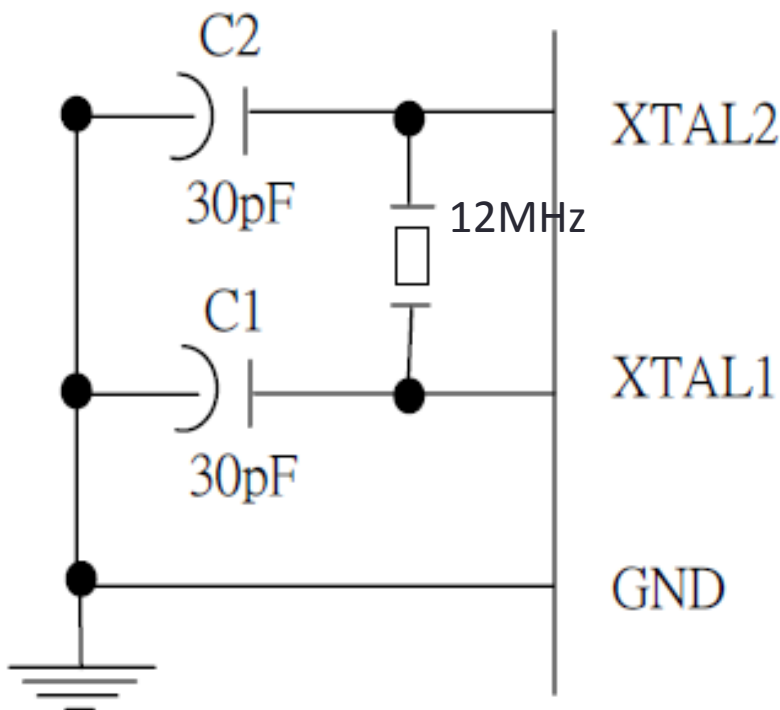
# XTAL1 and XTAL2

Crystal : XTAL •

• 8051 به کلاک خارجی نیاز دارد تا کار کند.

• یک اسیلاتور کریستال کوارتز باید به پایه‌های XTAL1 (پین ۱۹) و XTAL2 (پین ۱۸) متصل شود.

• کریستال 12MHz به همراه دو خازن 30pF



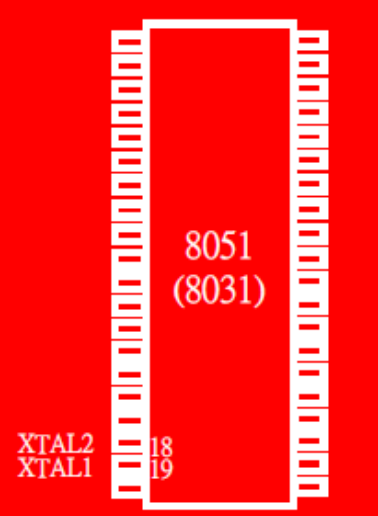
# XTAL1 and XTAL2

- عملیات CPU شامل ۴ فاز است که به آن سیکل ماشین گفته می‌شود.
  - واکنشی دستور از حافظه
  - دیکد کردن دستور
  - اجرای دستور
  - ذخیره نتایج
- هر کدام از فازها ممکن است یک یا چند کلاک به طول انجامد.
- هر دستور حداقل در یک سیکل ماشین اجرا می‌شود.
- اجرای بعضی دستورات پیچیده‌تر می‌توانند دو یا چند سیکل ماشین طول بکشد.
- هیچ دستوری مثلاً در ۱.۵ سیکل ماشین اجرا نمی‌شود.
- اگر دستوری مثلاً فاز ذخیره نتایج را نداشت، CPU باید به اندازه‌ی کلاک فاز ذخیره نتایج صبر کند و زمانی که یک سیکل ماشین کامل شد به اجرای دستور بعدی بپردازد.
- در 8051 هر سیکل ماشین ۱۲ کلاک اسیلاتور به طول می‌کشد.

8051  
(8031)

18  
19

XTAL2  
XTAL1



## XTAL1 and XTAL2

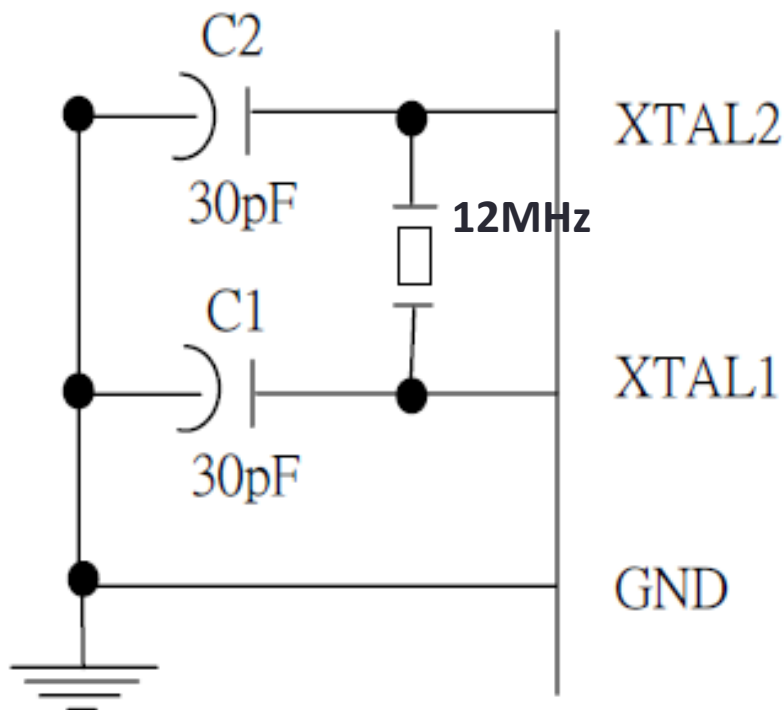
- در 8051 هر سیکل ماشین ۱۲ کلاک اسیلاتور به طول می کشد.
- یک دستور ساده، برای اجرا، به ۱۲ کلاک نیاز دارد.

• اگر کریستال دارای فرکانس 12MHz باشد،

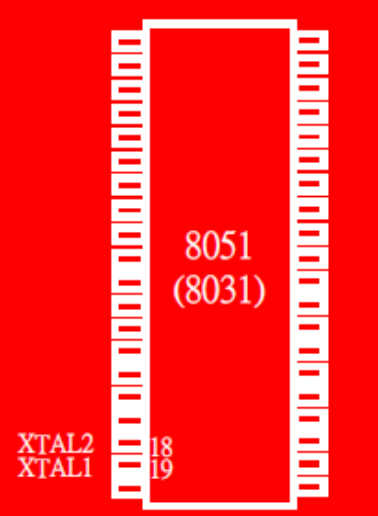
• زمان یک کلاک :  $1/12 \times 10^6 \text{ s} = 1/12 \mu\text{s}$

• زمان یک سیکل ماشین :  $12 \times 1/12 \mu\text{s} = 1 \mu\text{s}$

• یک دستور ساده در  $1 \mu\text{s}$  اجرا می شود.







## XTAL1 and XTAL2

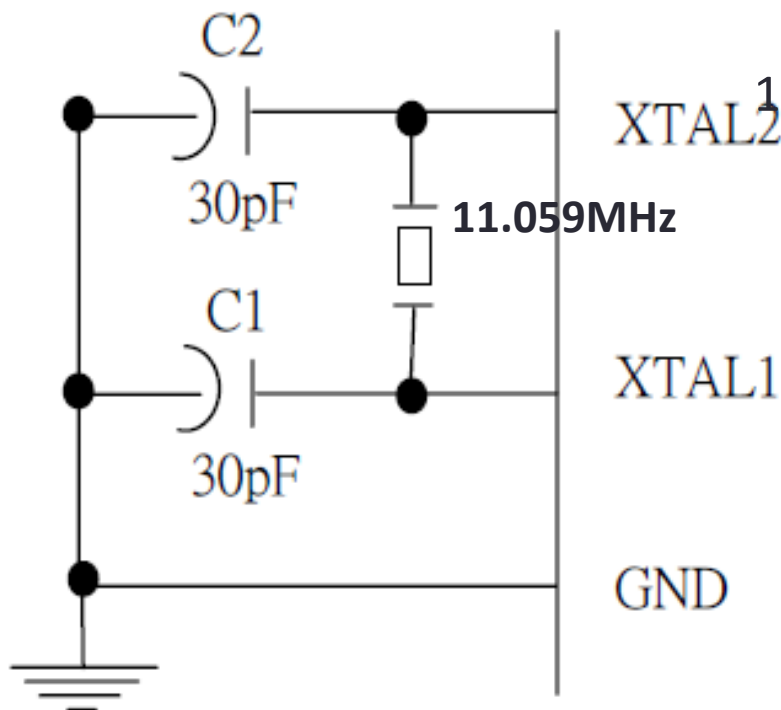
- در 8051 هر سیکل ماشین ۱۲ کلاک اسیلاتور به طول می کشد.
- یک دستور ساده، برای اجرا، به ۱۲ کلاک نیاز دارد.

• اگر کریستال دارای فرکانس 11.059MHz باشد،

• زمان یک کلاک :  $1/11.059 \times 10^6 \text{ s} = 0.0904 \mu\text{s}$

• زمان یک سیکل ماشین :  $12 \times 0.0904 \mu\text{s} = 1.085 \mu\text{s}$

• یک دستور ساده در  $1.085 \mu\text{s}$  اجرا می شود.



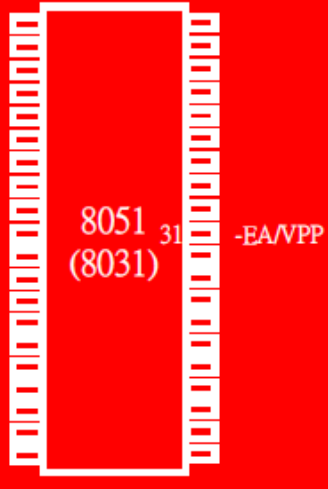
# RST

Reset : RST •

- زمانی که این ورودی برای حداقل دو سیکل ماشین برابر با یک باشد، میکروکنترلر ریست شده و تمام فعالیت‌ها از ابتدا شروع می‌شوند.
- یک سیکل ماشین در 8051 برابر با ۱۲ کلاک اسیلاتور است.
- اگر اسیلاتور 12MHz باشد، یک سیکل ماشین در 8051 برابر با ۱ میکرو ثانیه است.
- تمام ثبات‌های میکروکنترلر با مقادیر پیش فرض بار می‌شوند.

دوباره اولین خط برنامه در حافظه ROM اجرا خواهد شد.

Register	Reset Value
PC	0000
DPTR	0000
ACC	00
PSW	00
SP	07
B	00
P0-P3	FF



$\overline{EA}$

• External Access :  $\overline{EA}$

• این پایه باید به زمین متصل گردد تا فعال شود.

•  $\overline{EA}$  باید به VCC یا زمین متصل شود.

• اگر  $\overline{EA}$  به ۵ ولت متصل شود میکروکنترلر 8051 یا 8052 برنامه را از ROM داخلی اجرا می-کند.

• اگر EA به 0 متصل شود، ROM داخلی میکروکنترلر غیرفعال می‌شود و میکروکنترلر برنامه را فقط از ROM خارجی اجرا می‌کند.

• برای میکروکنترلرهای 8031 و 8032 که ROM داخلی ندارند،  $\overline{EA}$  باید به زمین متصل شود تا برنامه از حافظه‌ی خارجی اجرا شود.

# ALE و $\overline{\text{PSEN}}$

- Program Store Enable :  $\overline{\text{PSEN}}$
- Address Latch Enable : ALE

• از دو پین فوق عموماً در میکروکنترلر 8031 استفاده می‌شود.

•  $\overline{\text{PSEN}}$  حافظه ی برنامه خارجی را فعال می‌کند.

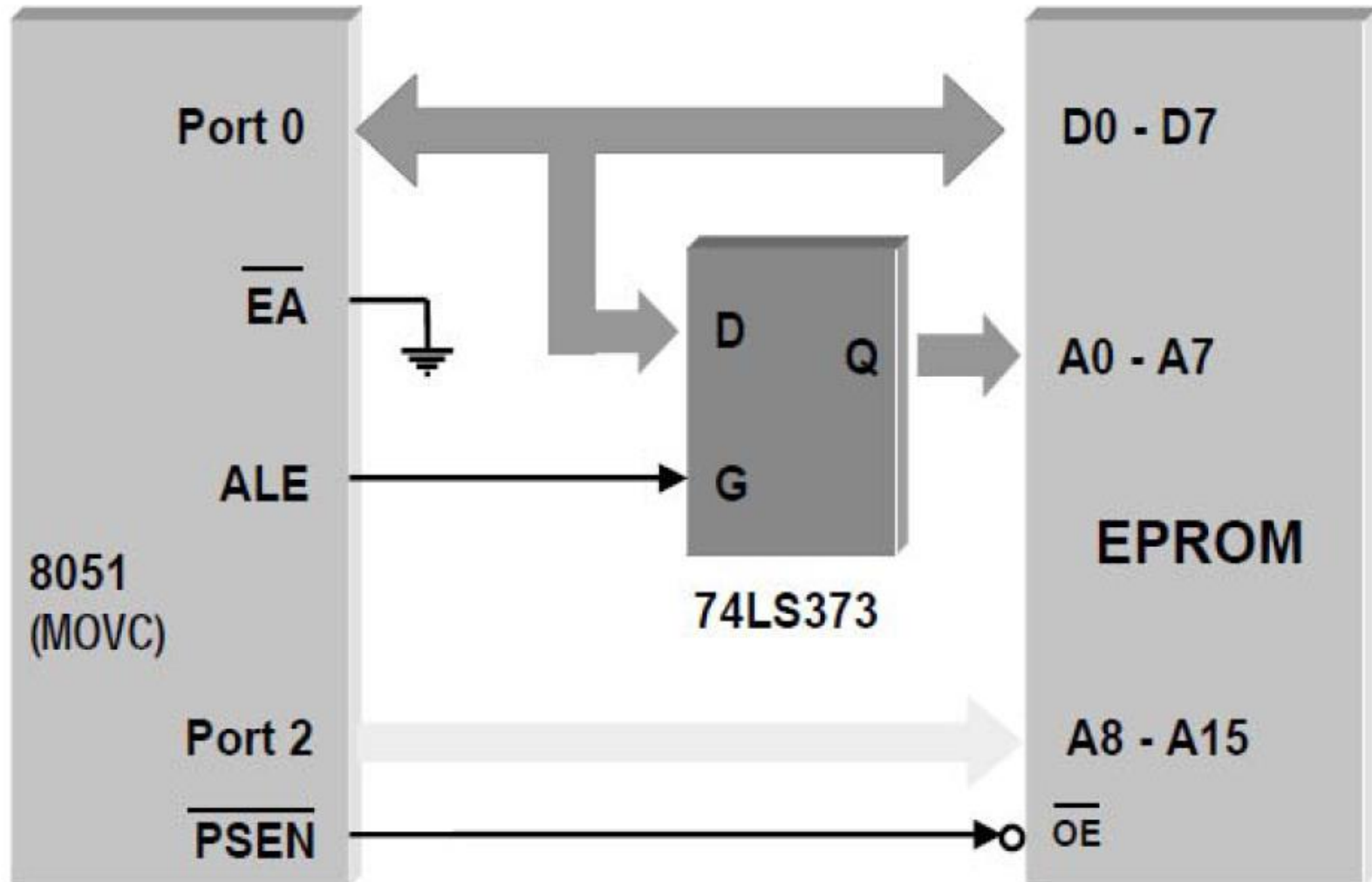
• این پین به پین  $\overline{\text{OE}}$  (output enable) حافظه ی خارجی وصل می‌شود.

• زمانی که برنامه از ROM داخلی اجرا می‌شود،  $\overline{\text{PSEN}}$  باید غیرفعال و برابر با ۱ باشد.

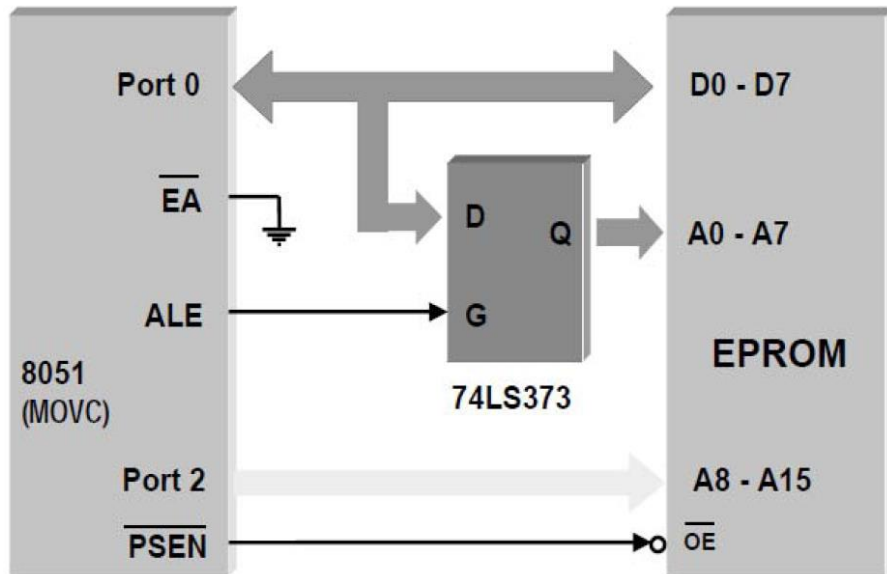
• در مواقعی که از حافظه ی خارجی استفاده می‌شود، پورت 0 هم برای آدرس و هم داده استفاده می‌شود و ALE برای جدا کردن (دی مالتی پلکس کردن) باس داده و آدرس بکار می‌رود.



# $\overline{\text{PSEN}}$ و ALE در فعال سازی حافظه کد خارجی



## $\overline{\text{PSEN}}$ و ALE در فعال سازی حافظه کد خارجی



- پورت صفر برای انتقال آدرس و داده

به صورت مالتی پلکس شده استفاده می شود.

- در نیمه اول سیکل حافظه، سیگنال ALE فعال می شود و باعث می شود ۸ بیت آدرس A0 تا A7 در رجیستری در حافظه خارجی ذخیره شود.

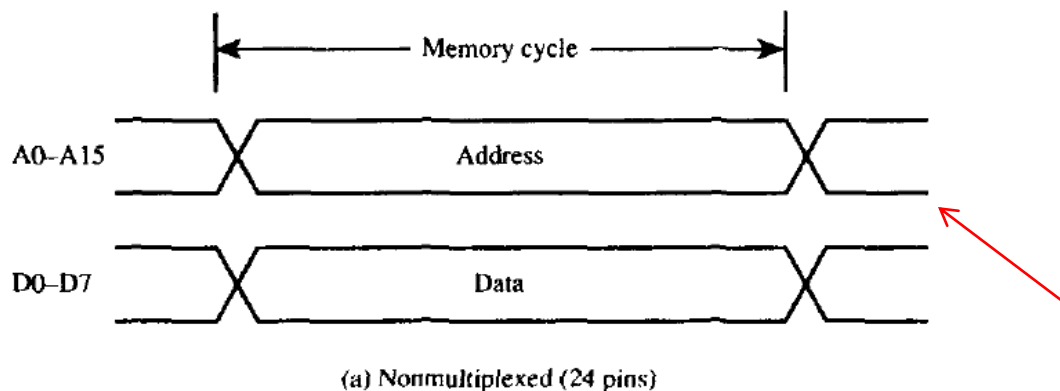
- در نیمه دوم سیکل حافظه ALE غیر فعال می شود و داده از طریق پورت صفر به میکروکنترلر انتقال می یابد.

- ۸ بیت بالاتر آدرس A8 تا A15 از طریق پورت ۲ فراهم می شود.

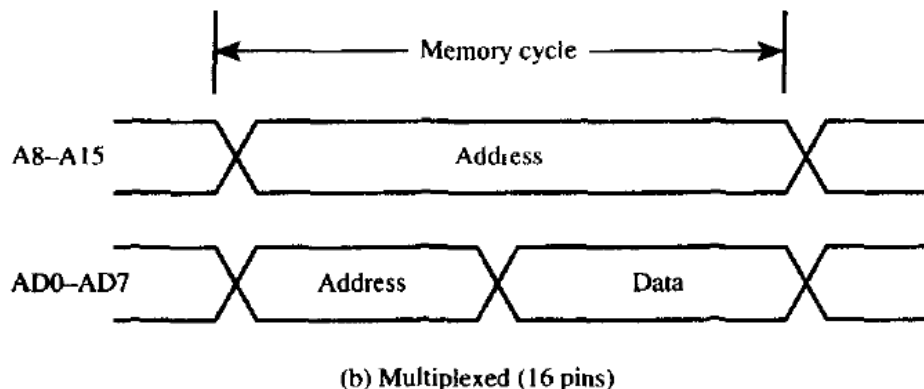
# PSEN و ALE در فعال سازی حافظه کد خارجی

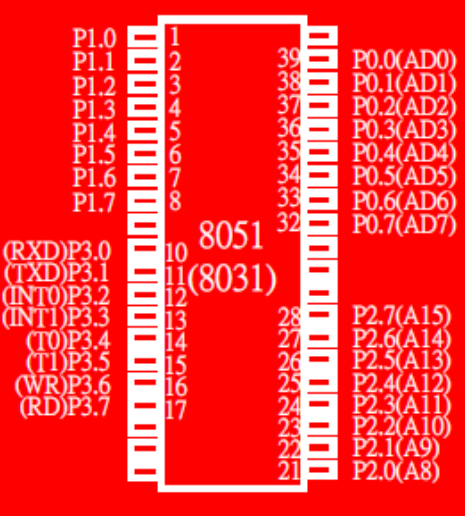
- برای اتصال حافظه‌ی کد خارجی 64k به میکروکنترلر به ۱۶ خط برای آدرس و ۸ خط برای داده نیاز داریم. یعنی جمعا ۲۴ خط.
- با روش مالتی پلکس کردن می‌توان از پورت صفر هم برای باس آدرس هم داده استفاده کنیم.

- تنها ۱۶ پایه درگیر اتصال به حافظه خارجی می‌شود.



- برای مثال P0 و P2 نیمه‌ی بالا و پایینی آدرس را مشخص کنند و P1 برای انتقال داده از حافظه‌ی خارجی به میکروکنترلر استفاده شود.





# پورت‌های I/O

• چهار پورت ۸ بیتی : P0, P1, P2, P3

- P0 و P2 برای اتصال حافظه‌ی خارجی نیز استفاده می‌شوند.
- P3 دارای پین‌ها با کاربرد خاص است.

• هنگام ریست شدن میکروکنترلر تمامی پورت‌ها مقدار FFH می‌گیرند. یعنی همه به عنوان ورودی خواهند شد.

- برای اینکه یک پورت به عنوان ورودی عمل کند باید ابتدا مقدار پین‌های آن همه یک باشد.

یک مقدار است نه  
شماره خانه حافظه

MOV A, #0FFH

FFH = 11111111

Accumulator

MOV P0, A

اگر عددی با حرف شروع شد، اول آن صفر قرار  
می‌دهیم تا با کاراکتر اشتباه نشود

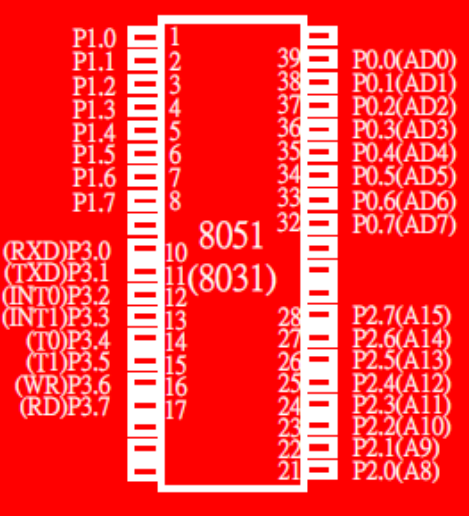
- برای اینکه یک پین پورت به عنوان ورودی عمل کند باید مقدار آن پین برابر با یک شود.

SETB P1.5

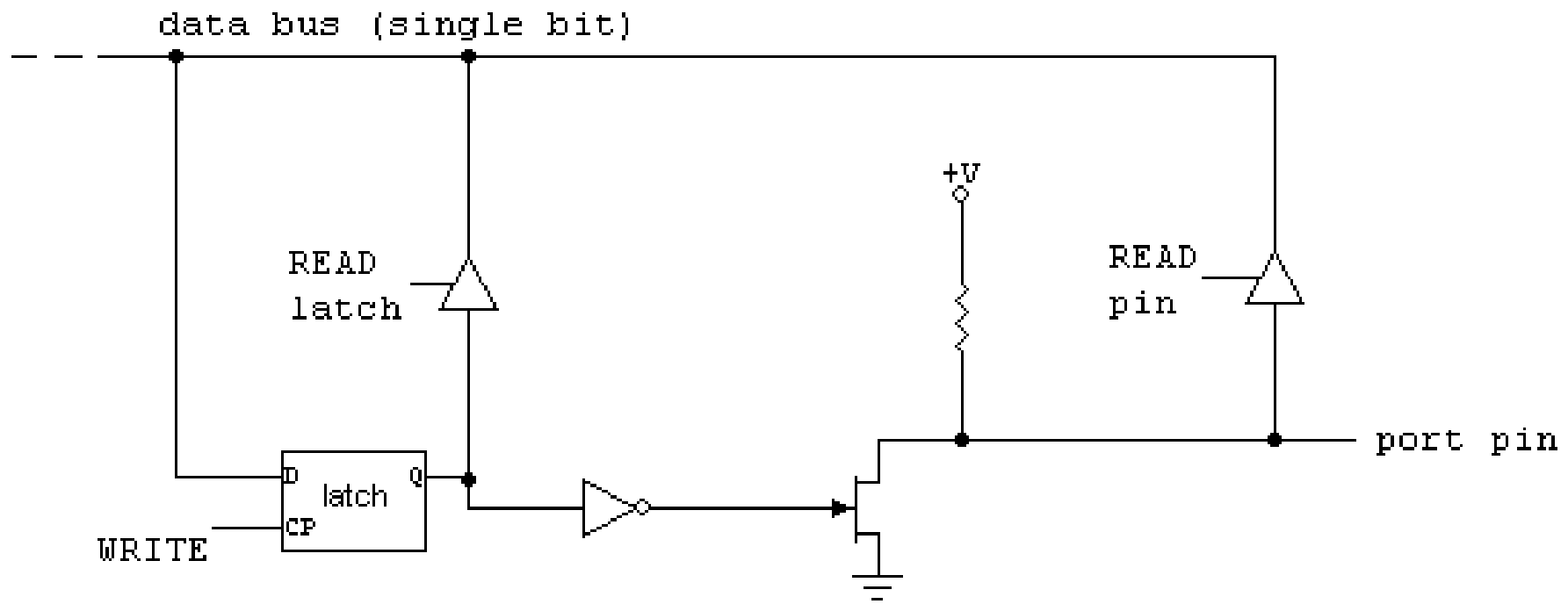
Set Bit

بیت پنجم پورت یک

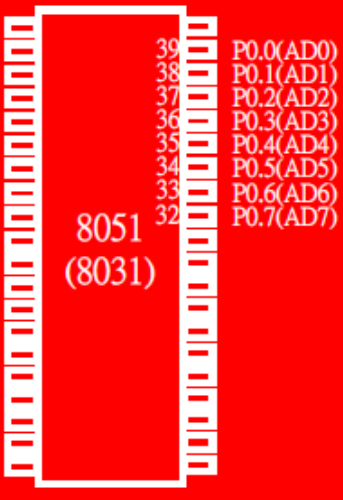




# ساختمان یک پایه ورودی خروجی 8051



8051 single port pin



## پورت صفر P0

• هنگامی که از حافظه‌ی خارجی استفاده نمی‌کنیم، این پورت برای عملیات ورودی-خروجی مورد استفاده قرار می‌گیرد.

• هنگامی که از حافظه‌ی خارجی استفاده می‌کنیم، این پورت به عنوان باس داده یا آدرس یا هر دو استفاده می‌شود.

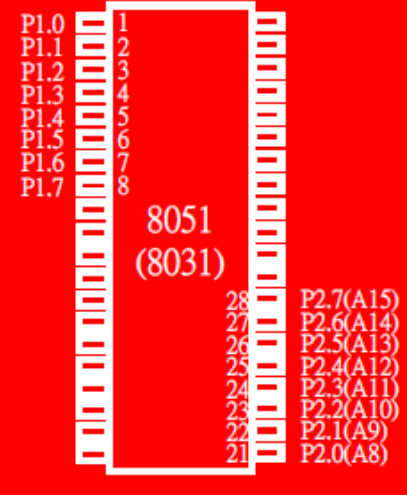
• ALE مشخص می‌کند پورت صفر باس داده است یا آدرس.

• زمانی که  $ALE=0$  است، P0 باس داده است D0-D7.

• زمانی که  $ALE=1$  است، P0 باس آدرس است A0-A7.

• این عمل برای صرفه‌جویی در پین‌های ورودی-خروجی انجام می‌شود.

## پورت P1 و P2



- پورت P1 همیشه برای ورودی-خروجی استفاده می‌شود.

- در میکروکنترلرهای 8032 و 8052 پایه‌های صفر و یک آن

(P1.1 و P1.0) می‌توانند به عنوان ورودی پالس خارجی تایمر سوم نیز به کار روند.

- پورت P2 هم به عنوان ورودی-خروجی و هم باس آدرس استفاده شود.

- هنگامی که حافظه داده یا کد خارجی بیش از ۲۵۶ بایت به میکروکنترلر متصل شود :

- P0 هشت بیت پایین باس آدرس را مشخص می‌کند A0-A7.

- P2 بیت‌های بالای باس آدرس را مشخص می‌کند A8-A15. در این حالت دیگر نمی‌تواند به عنوان ورودی-خروجی استفاده شود.

# پورت P3

• P3 به عنوان ورودی-خروجی استفاده می شود.

• پین های پورت P3 برای عملیات خاص نیز استفاده می شوند.



BIT	NAME	BIT ADDRESS	ALTERNATE FUNCTION	
P3.0	RXD	B0H	Receive data for serial port	ارتباطات سریال
P3.1	TXD	B1H	Transmit data for serial port	
P3.2	INT0	B2H	External interrupt 0	وقفه های خارجی
P3.3	INT1	B3H	External interrupt 1	
P3.4	T0	B4H	Timer/counter 0 external input	تایمرها
P3.5	T1	B5H	Timer/counter 1 external input	
P3.6	WR	B6H	External data memory write strobe	سیگنال های read و write برای حافظه داده خارجی
P3.7	RD	B7H	External data memory read strobe	

## تشکیلات حافظه

- 8051 حافظه‌ی مجزا برای برنامه و داده دارد.

- برنامه در ROM

- داده در RAM

- حافظه‌ی ROM و RAM ممکن است روی تراشه وجود داشته باشد.

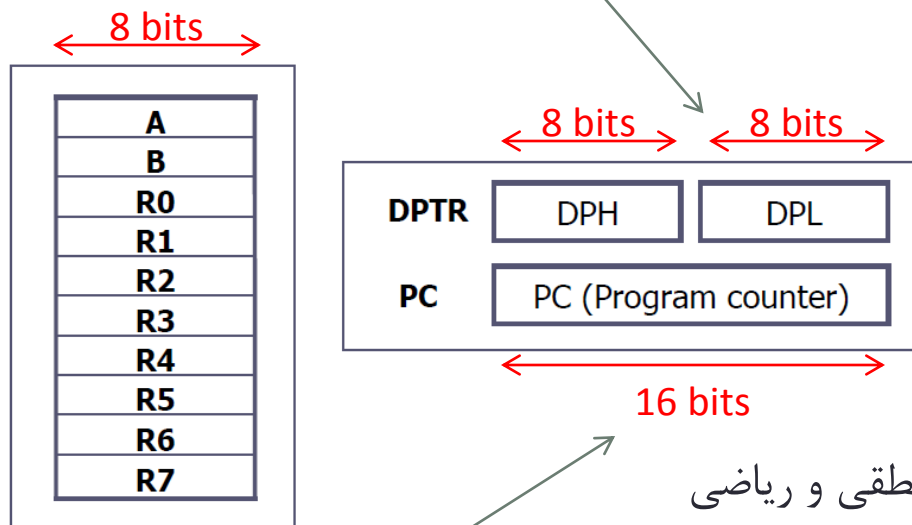
- امکان وصل کردن حافظه‌های خارجی برنامه تا حداکثر 64k بایت و حافظه داده خارجی تا حداکثر 64k بایت وجود دارد.

# رجیسترها

- رجیسترها برای ذخیره اطلاعات به صورت موقت استفاده می‌شوند.
- اطلاعات می‌تواند

- داده‌ای برای پردازش باشد.
- یا آدرس خانه‌ای از حافظه باشد.

برای اشاره به خانه‌های حافظه  
RAM خارجی که حداکثر 64k  
بایت است.



- اکثر رجیسترهای 8051 ۸ بیتی هستند.
- داده‌ها باید ۸ بیتی باشند.

- رجیسترهای پر استفاده در 8051 :

• A (accumulator) : برای تمامی دستورات منطقی و ریاضی

• B : برای کمک به A در اعمال ضرب و تقسیم

• R7, R6, R5, R4, R3, R2, R1, R0

• DPTR (data pointer) و PC (program counter)

برای اشاره به خانه‌های ROM  
داخلی که 4k بایت است یا ROM  
خارجی که حداکثر 64k بایت است.

# Numbers


Binary	Hexadecimal	Unsigned Decimal	Signed Decimal
10011010	9A	154	-102
00010001	11	17	17
11111111	FF	255	-1
01111111	7F	127	127
10000000	80	128	-128

# دستور MOV

## MOV destination, source

# مشخص می کند این یک مقدار است.

• مقدار مبدا را در مقصد کپی می کند.



```
MOV    A, #55H      ;load value 55H into reg. A
MOV    R0, A         ;copy contents of A into R0
                     ; (now A=R0=55H)
MOV    R1, A         ;copy contents of A into R1
                     ; (now A=R0=R1=55H)
MOV    R2, A         ;copy contents of A into R2
                     ; (now A=R0=R1=R2=55H)
MOV    R3, #95H      ;load value 95H into R3
                     ; (now R3=95H)
MOV    A, R3         ;copy contents of R3 into A
                     ; now A=R3=95H
```



# دستور MOV

- نکته : یک مقدار (که با # شروع می شود) می تواند به طور مستقیم در رجیسترهای A، B، R0 تا R7 بار شود.

MOV A, #23H

MOV R5, #0F9H

اگر # نباشد یعنی عددی که در خانه‌ی 23H حافظه است را داخل A بریز.

0 نشان می دهد که F یک عدد هگزادسیمال است نه یک کاراکتر.

- نکته : اگر مقدار 0 تا F را در یک رجیستر ۸ بیتی بریزیم، ۴ بیت بالایی صفر در نظر گرفته می شوند.

MOV A, #5

- در واقع این دستور برابر است با MOV A, #05. یعنی A=00000101

- ریختن عدد بزرگتر از سایز رجیستر تولید خطا می کند.

MOV A, #7F4H

- خطا :  $7F4 > FF$  (8bits)

# دستور ADD

ADD A, source

- مقدار مبدا را با مقدار accumulator جمع می کند و حاصل را در A می ریزد.
- مقدار مبدا می تواند یکی از رجیسترها و یا داده ی مستقیم باشد.
- مقصد همیشه A است.
- این دستورات نادرست اند : ADD R4, A و ADD R2, #12H

```
MOV A, #25H      ;load 25H into A
MOV R2, #34H     ;load 34H into R2
ADD A, R2 ;add R2 to Accumulator
               ; (A = A + R2)
```

```
MOV A, #25H      ;load one operand
               ;into A (A=25H)
ADD A, #34H      ;add the second
               ;operand 34H to A
```

# Program Counter

- PC به آدرس دستور بعدی که باید اجرا شود اشاره می کند.
- هر زمان که CPU دستور را از حافظه ی ROM واکشی کرد، مقدار PC اضافه می شود تا به آدرس دستور بعدی اشاره کند.
- PC ۱۶ بیتی است.
- به آدرس 0000H تا FFFFH می تواند اشاره کند.
- یعنی در کل 64k بایت برنامه.
- تمامی میکروکنترلرهای 8051 وقتی به برق متصل می شوند از خانه ی 0000 حافظه شروع به اجرای برنامه می نمایند.
- اولین کدی که در ROM ریخته می شود باید در خانه 0000H ریخته شود.
- برای این کار با دستور ORG در سورس برنامه، برای اسمبلر مشخص می کنیم اولین دستور را در کجای حافظه ی ROM قرار دهد.

# قرار دادن کد در ROM

شماره خط

comment

```

1 0000          ORG 0H          ;start (origin) at 0
2 0000  7D25     MOV R5,#25H    ;load 25H into R5
3 0002  7F34     MOV R7,#34H    ;load 34H into R7
4 0004  7400     MOV A,#0       ;load 0 into A
5 0006  2D       ADD A,R5       ;add contents of R5 to A
                                ;now A = A + R5
6 0007  2F       ADD A,R7      ;add contents of R7 to A
                                ;now A = A + R7
7 0008  2412     ADD A,#12H     ;add to A value 12H
                                ;now A = A + 12H
8 000A  80EF     HERE: SJMP HERE ;stay in this loop
9 000C          END            ;end of asm source file

```

label

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

## قرار دادن کد در ROM

- بعد از اینکه برنامه روی ROM ریخته شد، محتویات ROM بصورت زیر در خواهد آمد.

Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE



ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

## اجرای برنامه

Address	Code
→ 0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

- وقتی 8051 به برق وصل می‌شود، مقدار PC برابر با 0000 می‌شود.
- CPU کد موجود در خانه 0000 را واکنشی می‌کند.
- به محض اجرای کد 7D، CPU مقدار 25 را نیز از ROM واکنشی می‌کند و در R5 قرار می‌دهد.
- حال PC دو واحد زیاد شده و مقدارش 0002 می‌گردد.

## اجرای برنامه

Address	Code
0000	7D
0001	25
→ 0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

- به محض اجرای کد 7F، CPU مقدار 34 را نیز از ROM واکنشی می‌کند و در R7 قرار می‌دهد.
- حال PC دو واحد زیاد شده و مقدارش 0004 می‌گردد.

## اجرای برنامه

Address	Code
0000	7D
0001	25
0002	7F
0003	34
→ 0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

- دستور در آدرس 0004 اجرا می شود و مقدار A صفر می شود.
- مقدار PC برابر با 0006 می شود.



## اجرای برنامه

Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
→ 0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

- مقدار A برابر با 25 می شود.
- این دستور یک بایتی است و مقدار PC برابر با 0007 می شود.

## اجرای برنامه

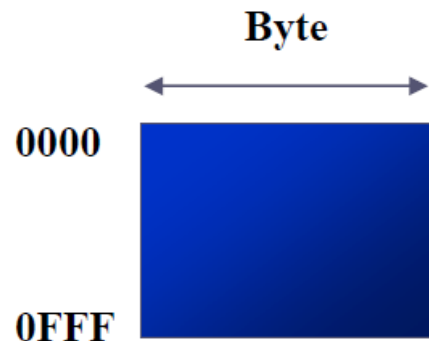
Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
→ 0007	2F
0008	24
0009	12
000A	80
000B	FE

ROM Address	Machine Language	Assembly Language
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	80EF	HERE: SJMP HERE

- برنامه به همین روال ادامه می یابد.

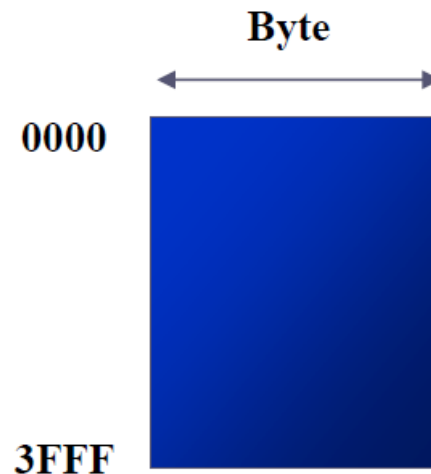
# حافظه ROM در خانواده 8051

- هیچ کدام از اعضای خانواده میکروکنترلرهای 8051 نمی‌تواند حافظه‌ی برنامه‌ی بزرگتر از 64k داشته باشد.
- زیرا PC ۱۶ بیتی است.



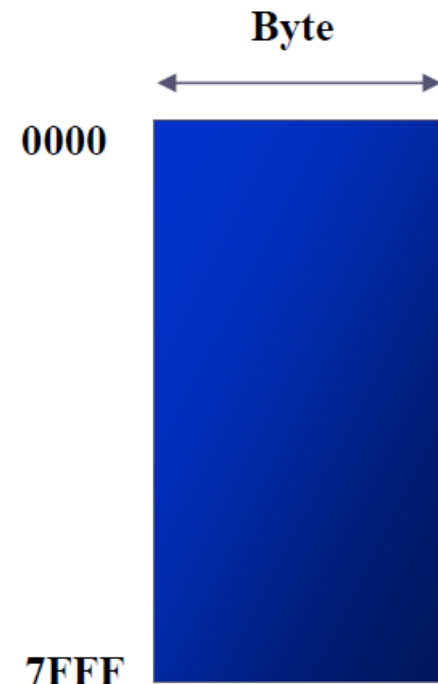
8751  
AT89C51

4k



DS89C420/30

16k




DS5000-32

32k

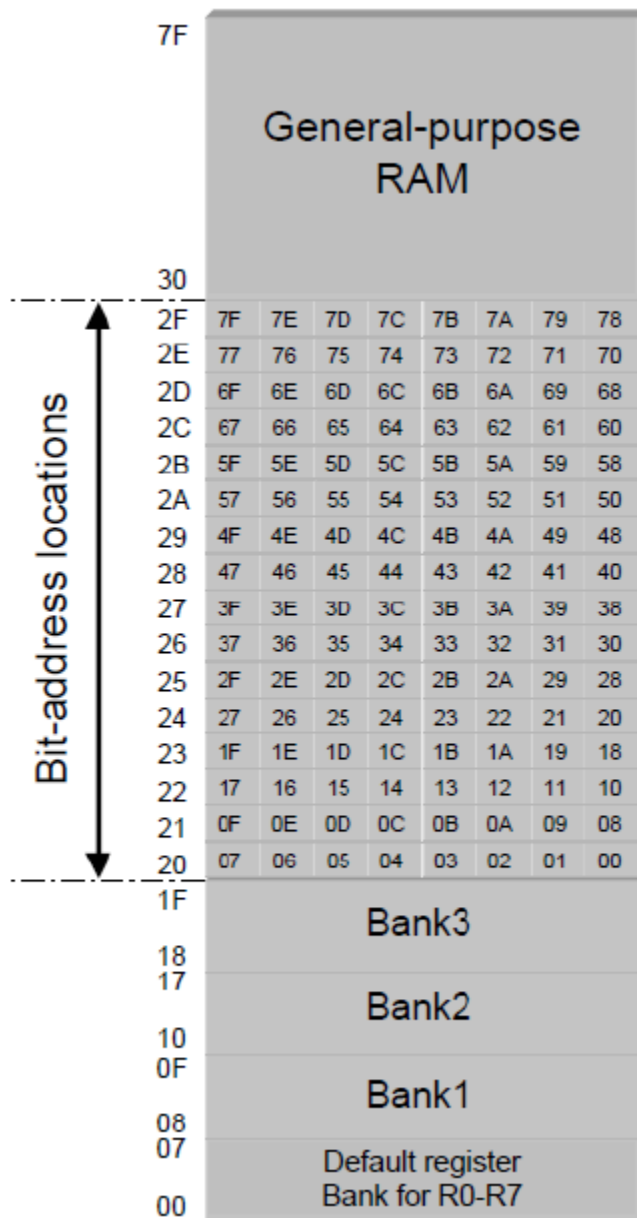
Size?

# Data Type

- میکروکنترلرهای 8051 تنها data type ۸ بیتی دارند.
- سائز رجیسترها هم ۸ بیتی است.
- اگر نیاز به کار کردن با داده‌های بیشتر از ۸ بیتی داشتیم، باید یک روشی برای این کار پیدا کنیم.
- تایپ داده‌ها می‌تواند مثبت یا منفی باشد.

- مثال  : چگونه دو عدد ۱۶ بیتی را با هم جمع کنیم؟  
 $2A46H + 3687H = ?$

## حافظه RAM

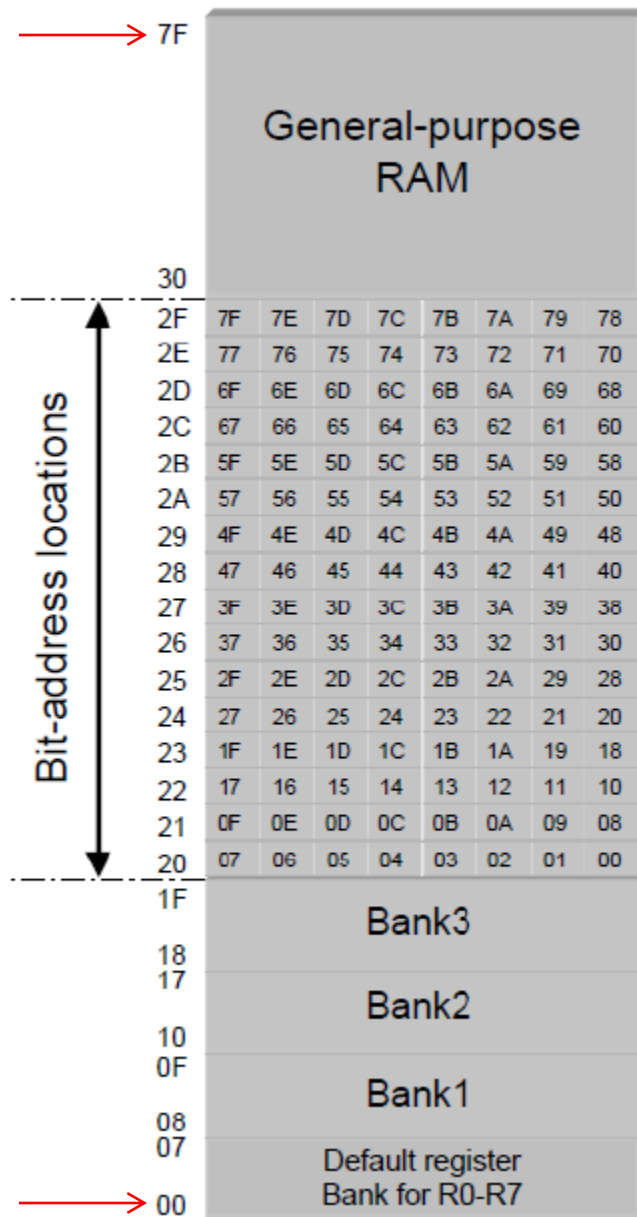


FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

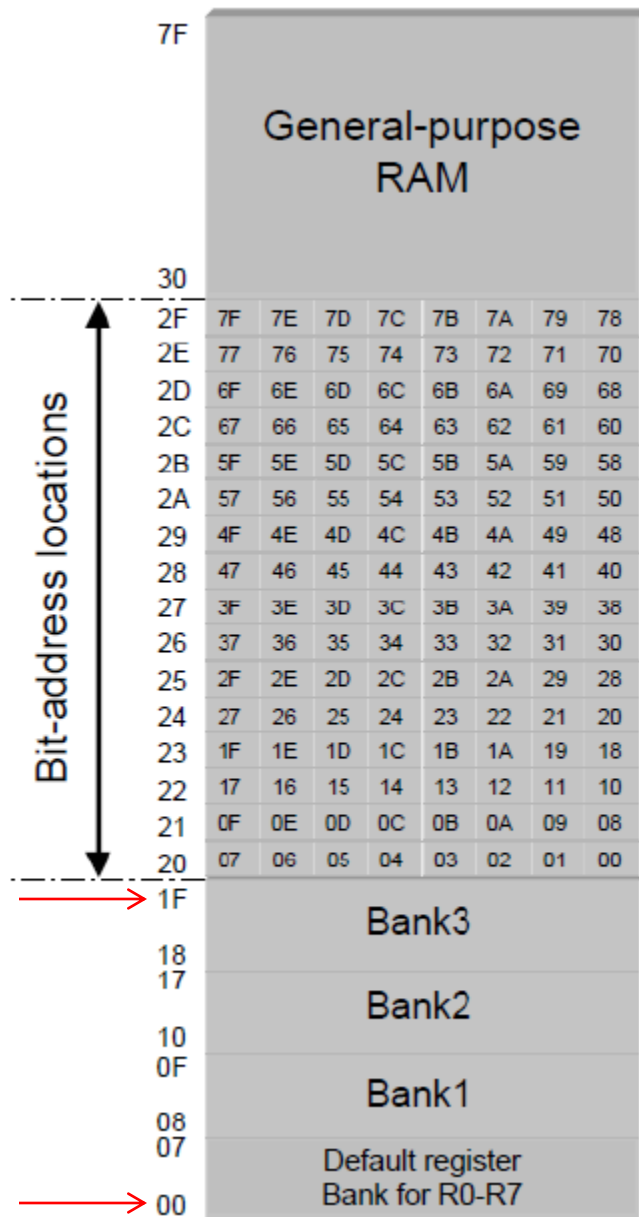
# حافظه RAM

- ۱۲۸ بایت RAM
- از آدرس ۰۰ تا ۷۷

- دارای سه بخش است :



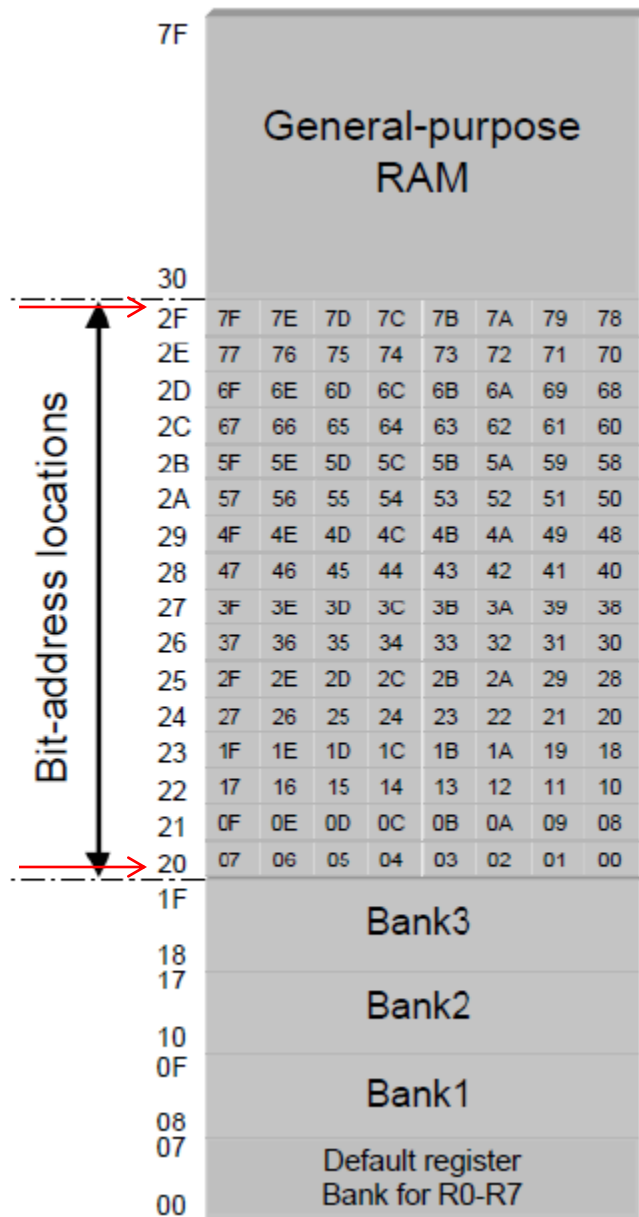
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0



## حافظه RAM

- ۱۲۸ بایت RAM
  - از آدرس 00 تا 7F
  - دارای سه بخش است :
۱. ۳۲ بایت از آدرس 00 تا 1F برای بانک رجیستر و پشته است.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

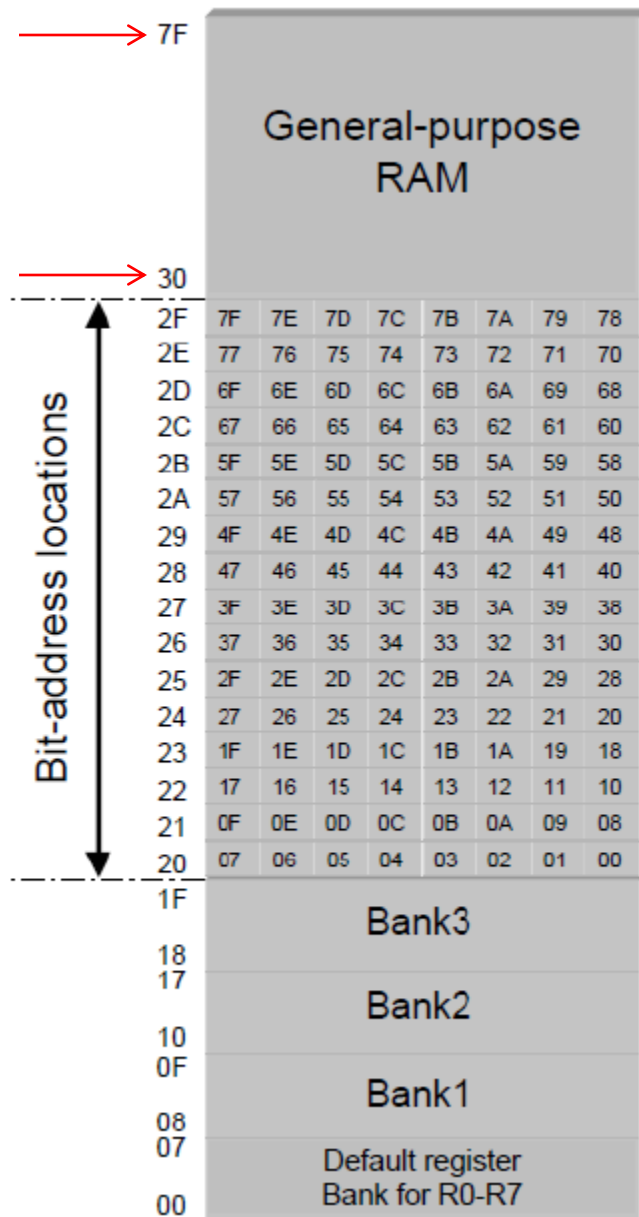


## حافظه RAM

- ۱۲۸ بایت RAM
  - از آدرس 00 تا 7F
  - دارای سه بخش است :
- ۲.
- ۱۶ بایت از آدرس 20 تا 2F حافظه با بیت آدرس پذیر است.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

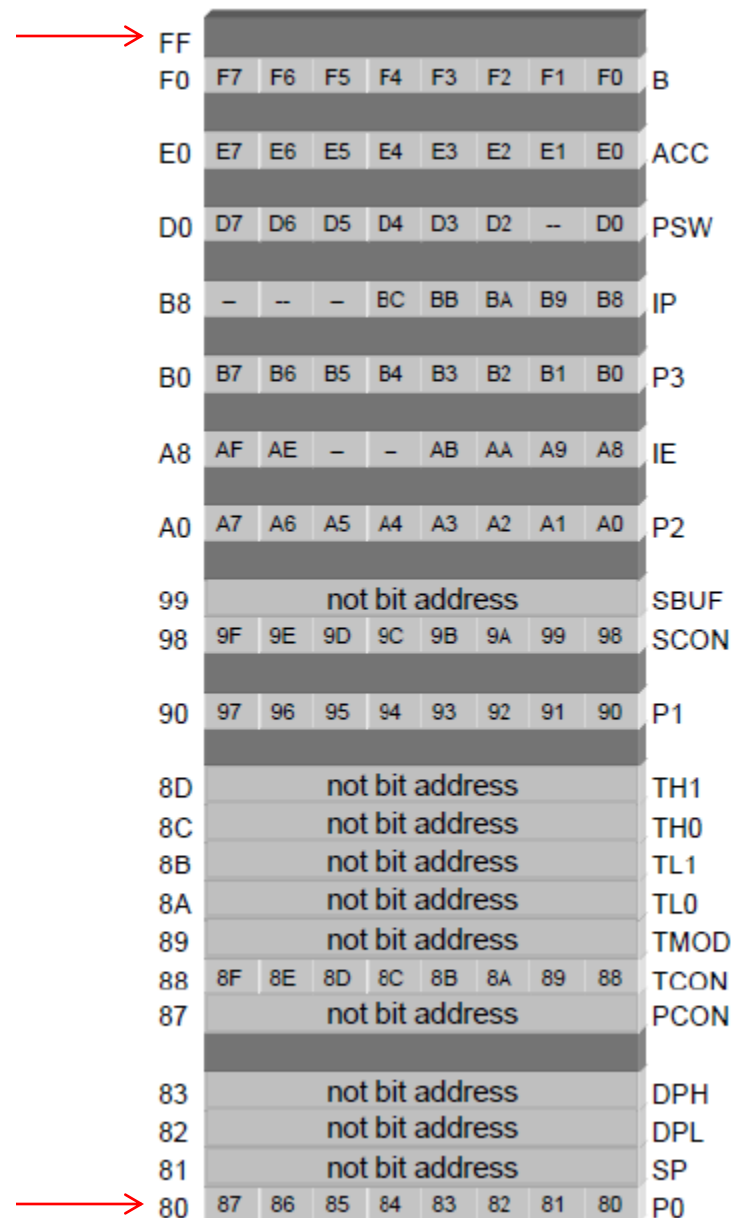
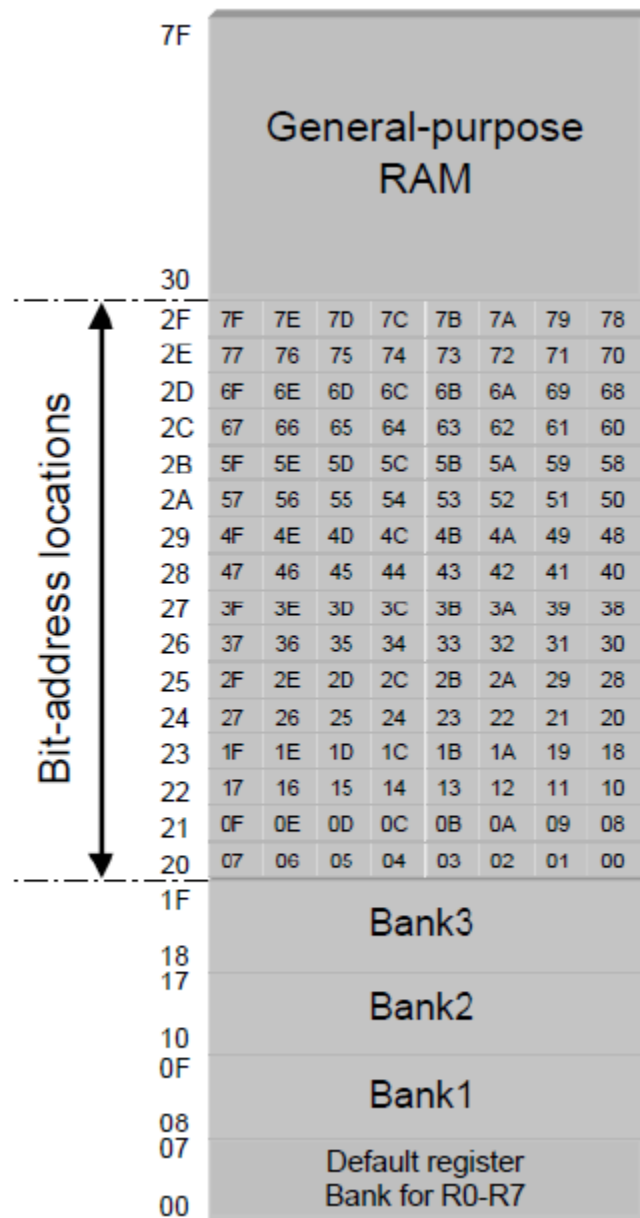




## حافظه RAM

- ۱۲۸ بیت RAM
- از آدرس 00 تا 7F
- دارای سه بخش است :
- ۳. ۸۰ بیت از آدرس 30 تا 7F برای استفاده‌ی عمومی از RAM است.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0



## حافظه RAM

• ۱۲۸ بیت RAM از آدرس 80 تا FF به ثباتها و پورتها اختصاص داده شده است.

• به این بخش ثباتهای کاربرد خاص (SFR) گفته می‌شود.

• SFR :

Special Function Register

SFR

# حافظه‌ی عمومی RAM

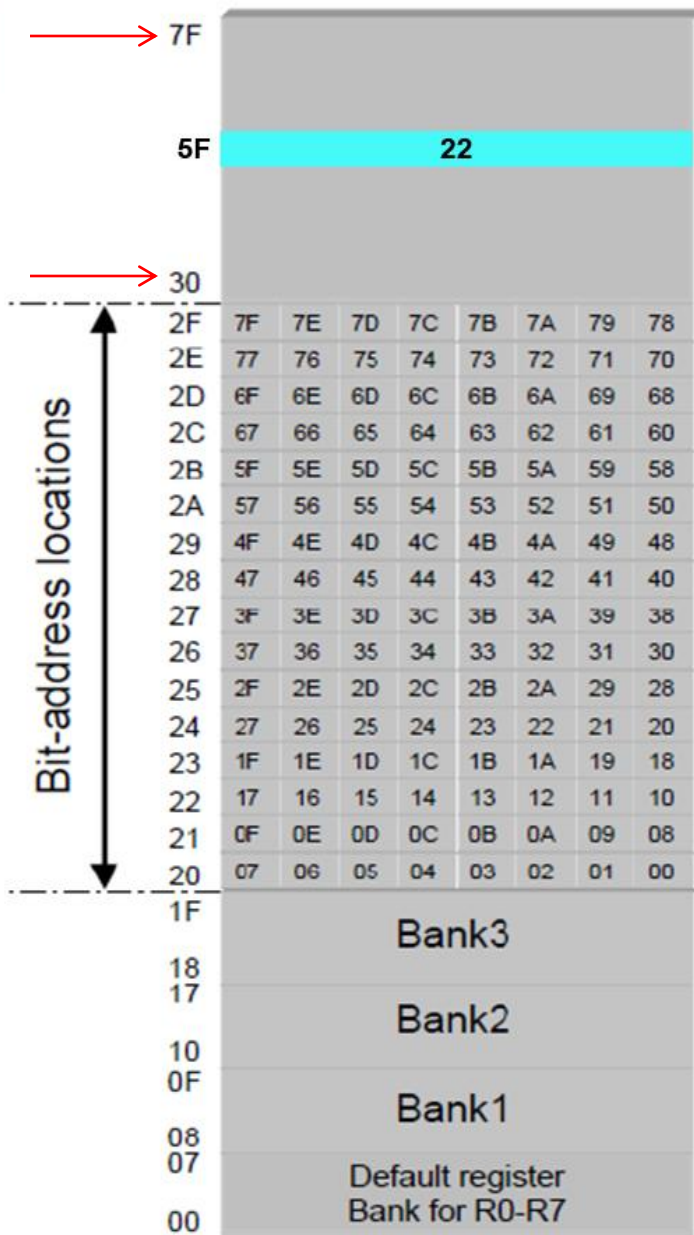
• ۸۰ بیت از آدرس 30 تا 7F

MOV A, #5FH

A = 5FH •

MOV A, 5FH

A = 22H •



# حافظه‌ی RAM با بیت آدرس پذیر

General-purpose RAM

7F

30

2F 7F 7E 7D 7C 7B 7A 79 78

2E 77 76 75 74 73 72 71 70

2D 6F 6E 6D 6C 6B 6A 69 68

2C 67 66 65 64 63 62 61 60

2B 5F 5E 5D 5C 5B 5A 59 58

2A 57 56 55 54 53 52 51 50

29 4F 4E 4D 4C 4B 4A 49 48

28 47 46 45 44 43 42 41 40

27 3F 3E 3D 3C 3B 3A 39 38

26 37 36 35 34 33 32 31 30

25 2F 2E 2D 2C 2B 2A 29 28

24 27 26 25 24 23 22 21 20

23 1F 1E 1D 1C 1B 1A 19 18

22 17 16 15 14 13 12 11 10

21 0F 0E 0D 0C 0B 0A 09 08

20 07 06 05 04 03 02 01 00

1F

Bank3

18

17

Bank2

10

0F

Bank1

08

07

Default register Bank for R0-R7

00

Bit-address locations

SETB 65H

• بیت 65H برابر با یک می شود.

Clear → CLR 65H

• بیت 65H برابر با صفر می شود.

MOV A, 2CH

Logical OR → ORL A, #00100000B

MOV 2CH, A

• بیت 65H برابر با یک می شود.

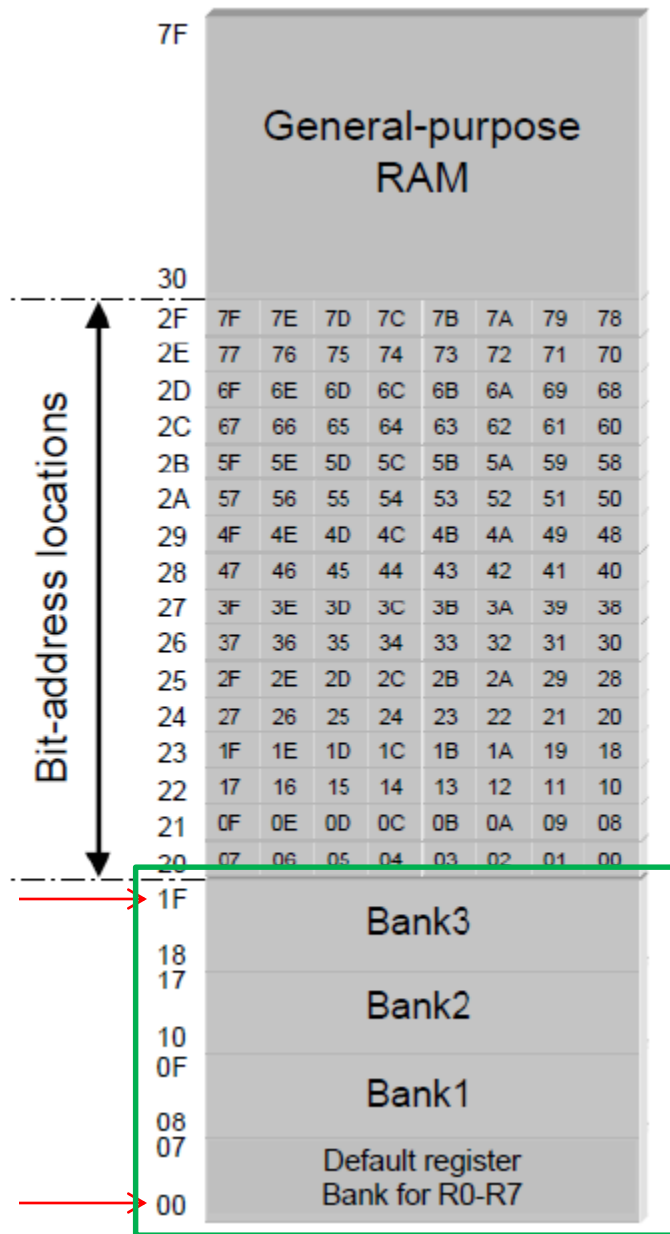
نشان می دهد عدد بصورت بیت است

• برای یک کردن بیت ششم خانه‌ی 3C در حافظه RAM از

کدام روش استفاده می شود؟

## بانک‌های ثابت

- چهار بانک ثابت داریم. در هر بانک ۸ رجیستر با نام‌های R0 تا R7 وجود دارد.



Bank 0	Bank 1	Bank 2	Bank 3
7 R7	F R7	17 R7	1F R7
6 R6	E R6	16 R6	1E R6
5 R5	D R5	15 R5	1D R5
4 R4	C R4	14 R4	1C R4
3 R3	B R3	13 R3	1B R3
2 R2	A R2	12 R2	1A R2
1 R1	9 R1	11 R1	19 R1
0 R0	8 R0	10 R0	18 R0

## بانک‌های ثبات

- در مواقعی که میکروکنترلر به برق متصل می‌شود یا ریست می‌شود، بانک صفر به طور پیش فرض فعال می‌شود.

• مثال : `MOV A, R5`

• مثال : `MOV A, 5`

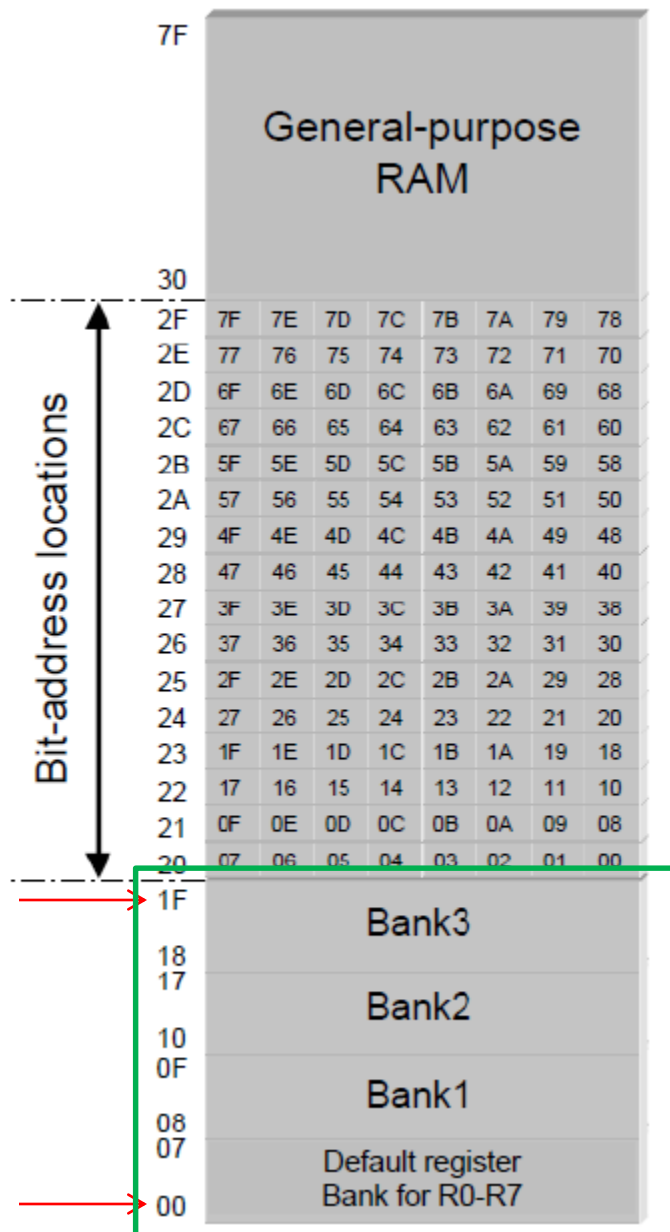
- مقدار خانه‌ی `05H` حافظه در داخل `A` قرار می‌گیرد.

- می‌توان با تغییر بیت‌های انتخاب ثبات در ثبات وضعیت، بانک‌های ۰ تا ۳ را فعال کرد.

- مثال : اگر بانک ثبات ۳ فعال باشد :

• `MOV A, R5`

- مقدار خانه‌ی `1DH` حافظه در داخل `A` قرار می‌گیرد.



## بانک‌های ثبات

- فایده : می‌توان بین بانک‌های مختلف به راحتی سوییچ کرد. می‌توان برای هر قسمت از برنامه بانک ثبات خاصی را در نظر گرفت.

MAIN:

.....	SUB1:	SUB2:
.....	.....	.....
MOV R0, #5	.....	.....
.....	MOV R0, #30	MOV R0, #22
.....	.....	.....
MOV R1, #40	.....	.....
.....	.....	MOV R1, #4
CALL SUB1	RET	.....
.....	.....	.....
.....	.....	.....
MOV A, R0	.....	RET
.....	.....	.....
.....	.....	.....
CALL SUB2	.....	.....
.....	.....	.....
MOV P0, R1	.....	.....
.....	.....	.....

- تابع MAIN در طول اجرا دو زیربرنامه SUB1 و SUB2 را فراخوانی می‌کند.
- ممکن است برنامه‌نویس در نوشتن زیربرنامه‌های SUB1 و SUB2 مقدار R0 یا R1 را تغییر دهد.
- حال زمانی که به برنامه‌ی اصلی باز می‌گردد، مقدار R0 و R1 تغییر یافته‌اند و این برخلاف انتظار است و ممکن است در روند اجرای برنامه خطا ایجاد شود.
- راه حل : به هر یک از قسمت‌های برنامه یک بانک ثبات اختصاص دهیم.

## بانک‌های ثبات

- فایده : می‌توان بین بانک‌های مختلف به راحتی سوییچ کرد. می‌توان برای هر قسمت از برنامه بانک ثبات خاصی را در نظر گرفت.

MAIN:

.....	SUB1:	SUB2:
.....	.....	.....
MOV R0, #5	.....	.....
.....	MOV R0, #30	MOV R0, #22
.....	.....	.....
MOV R1, #40	.....	.....
.....	MOV R1, #4	.....
CALL SUB1	.....	.....
.....	.....	.....
.....	.....	.....
MOV A, R0	.....	.....
.....	.....	.....
.....	.....	.....
CALL SUB2	.....	.....

بانک ثبات یک  
R0 تا R7 در خانه‌های 8  
تا F مقداردهی می‌شوند.

بانک ثبات دو  
R0 تا R7 در خانه‌های 10  
تا 18 مقداردهی می‌شوند.

بانک ثبات صفر  
R0 تا R7 در خانه‌های 0  
تا 7 مقداردهی می‌شوند.

- تابع MAIN در طول اجرا دو زیربرنامه SUB1 و SUB2 را فراخوانی می‌کند.
- ممکن است برنامه‌نویس در نوشتن زیربرنامه‌های SUB1 و SUB2 مقدار R0 یا R1 را تغییر دهد.
- حال زمانی که به برنامه‌ی اصلی باز می‌گردد، مقدار R0 و R1 تغییر یافته‌اند و این برخلاف انتظار است و ممکن است در روند اجرای برنامه خطا ایجاد شود.
- راه حل : به هر یک از قسمت‌های برنامه یک بانک ثبات اختصاص دهیم.



## ثبات‌های کاربرد خاص SFR

- علاوه بر ثبات‌های R0 تا R7، ۲۱ ثبات کاربرد خاص داریم.
- اکثر ثبات‌های خاص دارای آدرس مستقیم هستند. برخی از آنها دارای بیت‌های آدرس پذیر نیز می‌باشند.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

## ثبات‌های کاربرد خاص SFR

- علاوه بر ثبات‌های R0 تا R7، ۲۱ ثبات کاربرد خاص داریم.
- اکثر ثبات‌های خاص دارای آدرس مستقیم هستند. برخی از آنها دارای بیت‌های آدرس پذیر نیز می‌باشند.

MOV A, R0 = MOV 0E0H, R0

0 برای این است که مشخص کند E عدد هگزادسیمال است نه کاراکتر

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

## ثبات‌های کاربرد خاص SFR

- علاوه بر ثبات‌های R0 تا R7، ۲۱ ثبات کاربرد خاص داریم.
- اکثر ثبات‌های خاص دارای آدرس مستقیم هستند. برخی از آنها دارای بیت‌های آدرس پذیر نیز می‌باشند.

SETB 0F7H

- پرارزش ترین بیت اکومولاتور B را یک می‌کند.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

# ثبات وضعیت PSW

Program Status Word : PSW •

- شامل بیت‌های وضعیت میکروکنترلر است.
- این بیت‌ها پس از انجام هر دستور تنظیم می‌شوند.

PSW

C	AC	F0	RS1	RS0	OV	----	P
---	----	----	-----	-----	----	------	---

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

PSW

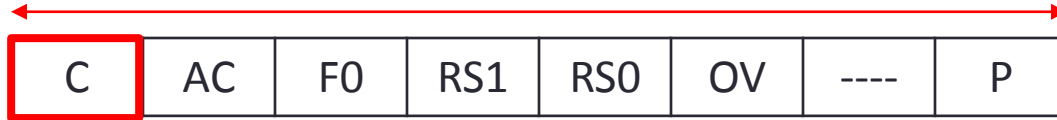


C	AC	F0	RS1	RS0	OV	----	P
---	----	----	-----	-----	----	------	---

## ثبات وضعیت PSW

توضیح		سمبول	بیت	آدرس
پرچم نقلی	Carry Flag	C یا CY	PSW.7	D7H
پرچم نقلی کمکی	Auxiliary Carry Flag	AC	PSW.6	D6H
پرچم صفر	Flag 0	F0	PSW.5	D5H
انتخاب بانک ثبات 1	Register Select 1	RS1	PSW.4	D4H
انتخاب بانک ثبات 0	Register Select 0	RS0	PSW.3	D3H
پرچم سرریز	Overflow Flag	OV	PSW.2	D2H
رزرو شده	Reserved	----	PSW.1	D1H
پرچم توازن زوج	Even Parity Flag	P	PSW.0	D0H

PSW



## بیت پرچم نقلی C

- در هنگام عملیات ADD اگر از بیت ۷ام بیت نقلی خارج شود، بیت پرچم نقلی C برابر با یک می‌شود.
- اگر در زمان تفریق از بیت ۸ قرض گرفته شود، این بیت برابر با یک می‌شود.
- مثال

```
MOV A, #0A0H
```

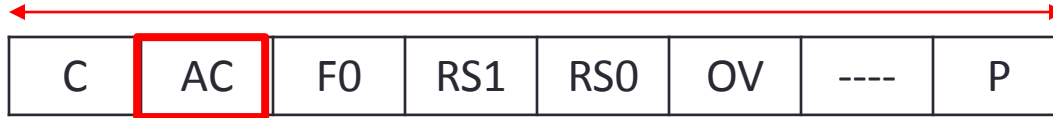
```
ADD A, #0AAH
```

- می‌توان بیت C را با دستوراتی به صورت مستقیم نیز تغییر داد.

```
ANL C, 25H
```

- مقدار C را با مقدار در بیت خانه 25H حافظه AND می‌کند.

PSW



## بیت نقلی کمکی AC

- در هنگام جمع دو عدد، اگر از بیت سوم رقم نقلی خارج شود، بیت AC برابر با یک می‌شود.
- در هنگام تفریق دو عدد، اگر از بیت چهارم قرض گرفته شود، بیت AC برابر با یک می‌شود.

	C	AC
	<span style="border: 1px solid red; padding: 2px;">1</span>	<span style="border: 1px solid red; padding: 2px;">0</span>
A5	10100101	
+89	+10001001	
<hr style="border-top: 1px dashed black;"/>		
2E	00101110	

	C	AC
	<span style="border: 1px solid red; padding: 2px;">1</span>	<span style="border: 1px solid red; padding: 2px;">1</span>
5D	01011101	
+C6	+11000110	
<hr style="border-top: 1px dashed black;"/>		
23	00100011	

	C	AC
	<span style="border: 1px solid red; padding: 2px;">1</span>	<span style="border: 1px solid red; padding: 2px;">0</span>
5D	01011101	
-C6	-11000110	
<hr style="border-top: 1px dashed black;"/>		
97	10010111	

PSW



# بیت پرچم F0

- این بیت برای استفاده‌ی کاربر آزاد است.



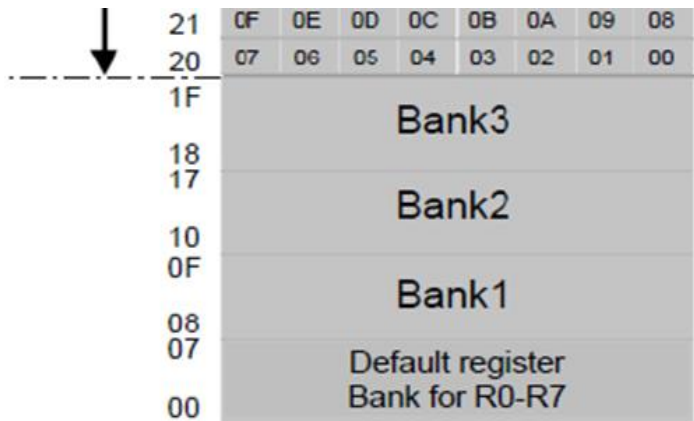
PSW



## بیت‌های انتخاب بانک ثبات

- با این دو بیت می‌توان هر یک از ۴ بانک ثبات را فعال نمود.

RS1	RS0	Register Bank	Address
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH



MOV R1, #40

- خانه 01H حافظه برابر عدد ۴۰ می‌شود.
- در حالت پیش فرض بانک ۰ فعال است.

PSW



## بیت‌های انتخاب بانک ثبات

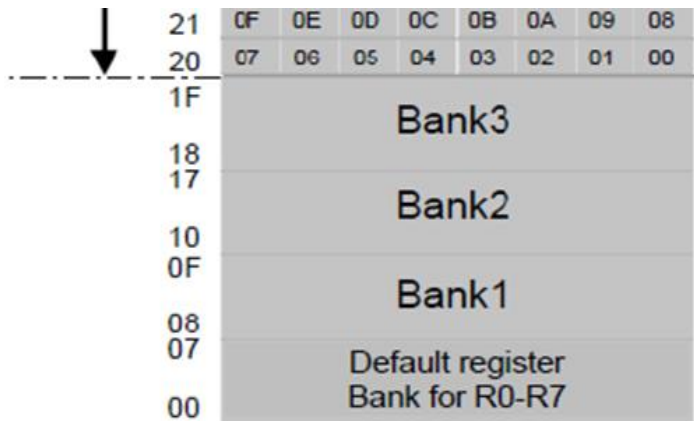
- با این دو بیت می‌توان هر یک از ۴ بانک ثبات را فعال نمود.

RS1	RS0	Register Bank	Address
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH

SETB RS1

CLR RS0

MOV R1, #40



- خانه 11H حافظه برابر عدد ۴۰ می‌شود.

PSW



## بیت‌های انتخاب بانک ثبات

- با این دو بیت می‌توان هر یک از ۴ بانک ثبات را فعال نمود.

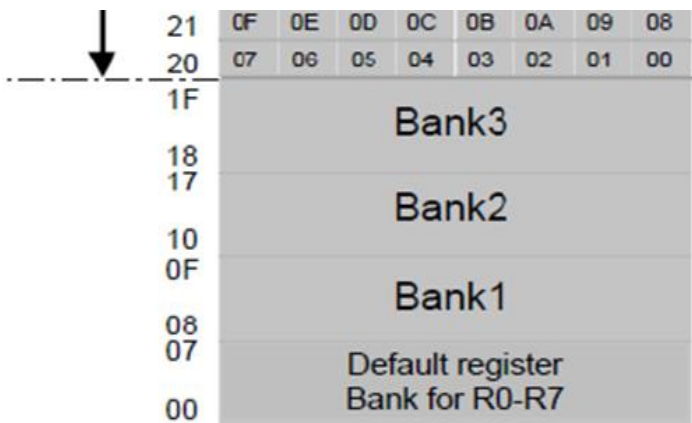
RS1	RS0	Register Bank	Address
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	1	3	18H – 1FH

SETB PSW.4

SETB PSW.3

MOV R1, #40

- خانه 19H حافظه برابر عدد ۴۰ می‌شود.



PSW



## بیت‌های انتخاب بانک ثبات

• دستورات زیر چه تغییراتی در خانه‌های RAM ایجاد می‌کنند؟

21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F	Bank3							
18	Bank2							
17								
10	Bank1							
0F								
08	Default register Bank for R0-R7							
07								
00								

MOV R1, #22

MOV R7, #30

MOV 8, #40

SETB RS1

MOV R1, #14

MOV R0, 1

MOV A, R1

CLR RS1

SETB RS0

MOV R1, A

MOV 19, #3

## بیت پرچم سرریز OV

- بعد از عملیات جمع یا تفریق اعداد علامت‌دار در صورتی که سرریز رخ دهد این بیت برابر با یک می‌شود.
- در موقع جمع دو عدد علامت‌دار اگر نتیجه بزرگتر از ۱۲۷ یا کوچکتر از ۱۲۸- باشد، سرریز اتفاق افتاده و نتیجه معتبر نیست.
- حاصل جمع دو عدد مثبت، منفی یا حاصل جمع دو عدد منفی، مثبت شود.

$$(+A) + (+B) = (-C)$$

$$(-A) + (-B) = (+C)$$

- در تفریق دو عدد علامت‌دار نیز اگر اتفاقات زیر رخ دهد سرریز رخ داده است.

$$(+A) - (-B) = (-C)$$

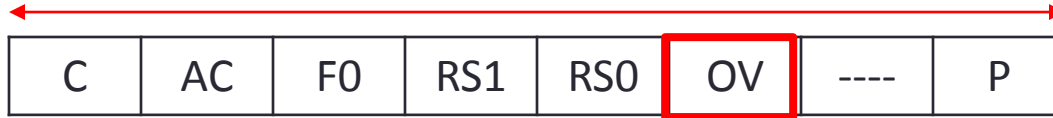
$$(-A) - (+B) = (+C)$$

## بیت پرچم سرریز OV

- در موقع جمع دو عدد علامت‌دار اگر نتیجه بزرگتر از ۱۲۷ یا کوچکتر از ۱۲۸- باشد، سرریز اتفاق افتاده و نتیجه معتبر نیست.
- یا به عبارتی حاصل جمع دو عدد مثبت، منفی یا حاصل جمع دو عدد منفی، مثبت شود.

Binary	Unsigned Decimal	Signed Decimal	Hexadecimal
00001111	15	15	0F
+ <u>01111111</u>	127	127	7F
10001110	142	-114 !!!	8E

PSW



# بیت پرچم سرریز OV

- در کار با اعداد دهدهی بدون علامت به C نگاه می کنیم.
- در کار با اعداد دهدهی علامت دار به OV نگاه می کنیم.

Binary	Unsigned Decimal	Signed Decimal	Hexadecimal
10011101	157	-99	9D
+ <u>11100010</u>	226	-30	E2
<b>01111111</b>	<b>127</b>	<b>127 !!!</b>	<b>7F</b>

• مثال :

MOV R7, #0FFH

MOV A, #0FH

ADD A, R7

- R7=-1 و A=15 و جواب برابر 14 است. پس سرریز رخ نداده است.

PSW

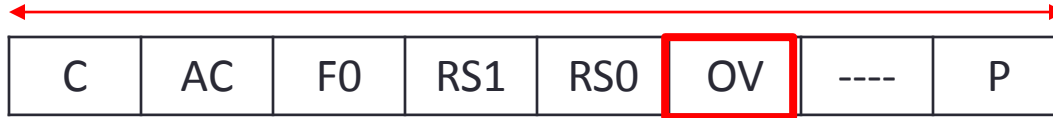


## بیت پرچم سرریز OV

- برای بررسی وقوع سرریز در زمان جمع باید به رقم‌های نقلی آخر و یکی مانده به آخر نگاه کنیم.
- اگر یکی از این دو رقم نقلی یک و دیگر صفر بود سرریز رخ داده و OV یک می‌شود.
- برای بررسی وقوع سرریز در زمان تفریق باید به بیت‌های آخر و یکی مانده به آخر نگاه کنیم.
- اگر یکی از این دو بیت رقم نقلی قرض گرفته باشد و دیگری قرض نگرفته باشد سرریز رخ داده و OV یک می‌شود.



PSW



# بیت پرچم سرریز OV

- در کار با اعداد دهدهی بدون علامت به C نگاه می‌کنیم.
- در کار با اعداد دهدهی علامت‌دار به OV نگاه می‌کنیم.

C=1  
OV=0

Binary	Unsigned Decimal	Signed Decimal	Hexadecimal
00010110	22	22	16
- 10110000	176	-80	B0
01100110	102	102	66

C=0  
OV=1

10011010	154	-102	9A
- 01110000	112	112	70
00101010	42	42	2A

C=1  
OV=1

01000000	64	64	40
- 10000000	128	-128	80
11000000	192	-64	C0

## Instructions that affect flag bits

Instruction	CY	OV	AC
ADD	X	X	X
ADDC	X	X	X
SUBB	X	X	X
MUL	0	X	
DIV	0	X	
DA	X		
RPC	X		
PLC	X		
SETB C	1		
CLR C	0		
CPL C	X		
ANL C, bit	X		
ANL C, /bit	X		
ORL C, bit	X		
ORL C, /bit	X		
MOV C, bit	X		
CJNE	X		

دستوراتی که بر روی بیت‌های  
پرچم تاثیر گذارند

PSW



## بیت توازن P

- این بیت به صورتی تغییر می‌کند که تعداد بیت‌های یک A به علاوه‌ی بیت P همواره زوج باشد.

MOV A, #55H

- P=0 خواهد شد.

## مثال

- پرچم‌های نقلی و نقلی کمکی و توازن بعد از انجام عملیات زیر چیست؟

MOV A, #38H

ADD A, #2FH

38	00111000
+ 2F	<u>00101111</u>
67	01100111

C = 0

AC = 1

P = 1

OV = 0

## مثال

- پرچم‌های نقلی و نقلی کمکی و توازن بعد از انجام عملیات زیر چیست؟

MOV A, #9CH

ADD A, #64H

$$\begin{array}{r}
 9C \quad 10011100 \\
 + \quad 64 \quad 01100100 \\
 \hline
 100 \quad 00000000
 \end{array}$$

C = 1

AC = 1

P = 0

OV = 0

## پشته Stack

- قسمتی از حافظه RAM است که برای نگهداری موقت داده یا آدرس استفاده می‌شود.
- رجیستر SP (Stack Pointer) برای اشاره به پشته استفاده می‌شود.

## اشاره گر پشته SP

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPI
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

- SP آدرس خانه‌ی بالای پشته را نشان می‌دهد.
- اطلاعات با دستور PUSH داخل پشته قرار می‌گیرد.
- با دستور POP اطلاعات از خانه بالای پشته خوانده می‌شود.

• PUSH :

- ابتدا یک واحد به SP اضافه می‌شود.
- اطلاعات در خانه‌ی حافظه‌ای که SP به آن اشاره می‌کند قرار گرفته می‌شود.

• POP :

- ابتدا اطلاعات از خانه‌ای که SP به آن اشاره می‌کند خوانده می‌شود.

- مقدار SP یک واحد کم می‌شود.

7F	General-purpose RAM							
30								
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
29	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F	Bank3							
18								
17	Bank2							
10								
0F	Bank1							
08								
07	Default register							
00	Bank for R0-R7							

## اشاره گر پشته SP

- مثال : می‌خواهیم پشته از آدرس 60H حافظه شروع شود.

MOV SP,? #5FH

- فضای پشته؟

## 32 Byte

- وقتی میکروکنترلر ریست می‌شود، مقدار SP برابر با 07H قرار می‌گیرد. یعنی اولین خانه‌ی پشته خانه‌ی 08H RAM خواهد بود.
- دستور فراخوانی زیربرنامه CALL به طور خودکار مقدار PC را در حافظه پشته ذخیره و دستور بازگشت از زیربرنامه RET آدرس برگشت را از پشته خوانده و در PC قرار می‌دهد.



# مثال

• فرض کنیم SP به صورت خودکار مقداردهی شده باشد. SP=07H

```
MOV R6, #25H
MOV R1, #12H
MOV R4, #0F3H
PUSH 6
PUSH 1
PUSH 4
```

آدرس دهی مستقیم

**Solution:**

	After PUSH 6	After PUSH 1	After PUSH 4
0B			
0A			F3
09		12	12
08	25	25	25
Start SP = 07	SP = 08	SP = 09	SP = 0A

## مثال

• تمام مقادیر HEX هستند.

```
POP      3      ; POP stack into R3
POP      5      ; POP stack into R5
POP      2      ; POP stack into R2
```

**Solution:**

		After POP 3	After POP 5	After POP 2
0B	54	0B	0B	0B
0A	F9	0A	0A	0A
09	76	09	09	09
08	6C	08	08	08
Start SP = 0B		SP = 0A	SP = 09	SP = 08

Because locations 20-2FH of RAM are reserved for bit-addressable memory, so we can change the SP to other RAM location by using the instruction "MOV SP, #XX"

## نکاتی در مورد پشته

- فضای پشته از پایین به بالا در RAM رشد می کند.
- بررسی شود که فضای پشته به انتهای RAM نرسد.
- هنگام ریست شدن میکروکنترلر، پشته و بانک یک ثبات‌ها از یک حافظه استفاده می نمایند.
- باید توجه شود برخورد پیش نیاید.
- می توان پشته را با دستور `MOV SP,#x` به جای دیگری از RAM منتقل کرد.

## مثال

```

MOV SP, #5FH    ;make RAM location 60H
                  ;first stack location

MOV R2, #25H
MOV R1, #12H
MOV R4, #0F3H
PUSH 2
PUSH 1
PUSH 4

```

**Solution:**

		After PUSH 2	After PUSH 1	After PUSH 4
63		63	63	63
62		62	62	62 F3
61		61	61 12	61 12
60		60 25	60 25	60 25
Start SP = 5F		SP = 60	SP = 61	SP = 62

# اشاره گر داده DPTR

## Data Pointer

• یک ثابت ۱۶ بیتی است.

• بایت کم ارزش تر DPL

• بایت پر ارزش تر DPH

• برای دسترسی به حافظه کد و برنامه‌ی خارجی استفاده می‌شود.

• مثال : دستورات زیر عدد 55H را در خانه 1000H حافظه داده خارجی می‌ریزد.

MOV A, #55H

MOV DPTR, #1000H ← 16bits DPL = 00H

DPH=10H

MOVX @DPTR, A

Move External

آدرس دهی غیرمستقیم

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

## ثبات‌های پورت

- پورت‌ها برای انتقال ورودی خروجی به کار می‌روند.
- مثال : اگر یک LED به بیت ۷ پورت یک متصل باشد،

SETB P1.7

- LED را روشن می‌کند.

CLR 97H یا CLR P1.7

- LED را خاموش می‌کند.

- اگر بخواهیم پورت به صورت ورودی باشد، باید ابتدا مقدار آن یک شود.

MOV P0,#0FFH  
SETB P3.1

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	-	-	-	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	-	-	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

## ثبات‌های تایمر

• 8051 دارای ۲ تایمر ۱۶ بیتی به نام‌های تایمر ۰ و تایمر ۱ است.

• برای اندازه‌گیری زمان و شمارش رویداد استفاده می‌شوند.

• بایت کم ارزش تر تایمر ۰  $TL0 = 0$

• بایت پر ارزش تر تایمر ۰  $TH0 = 0$

• بایت کم ارزش تر تایمر ۱  $TL1 = 1$

• بایت پر ارزش تر تایمر ۱  $TH1 = 1$

• طرز کار تایمر در ثبات حالت تایمر TMOD و ثبات کنترل TCON مشخص می‌شود.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

## ثبات‌های پورت سری

• 8051 دارای یک پورت سری است که برای ارسال و دریافت اطلاعات به صورت سریال استفاده می‌شود.

• تنظیمات پورت سری در ثبات SCON انجام می‌شود.

• اطلاعات ارسالی یا دریافتی در ثبات بافر پورت سری SBUF قرار می‌گیرند.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0



## ثبات‌های وقفه

- 8051 دارای ۶ منبع وقفه با دو سطح اولویت است.
- بعد از ریست شدن وقفه‌ها غیرفعال می‌شوند.
- با استفاده از ثبات فعال ساز وقفه IE (Interrupt Enable) می‌توان آنها را فعال نمود.
- با استفاده از ثبات اولویت وقفه IP (Interrupt Priority) اولیت وقفه‌ها مشخص می‌شوند.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	-	-	-	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	-	-	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0

# ثبات کنترل توان

Power Control •

• بیت‌های کنترلی مختلفی دارد.

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	--	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	AE	--	--	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit address								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit address								TH1
8C	not bit address								TH0
8B	not bit address								TL1
8A	not bit address								TL0
89	not bit address								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit address								PCON
83	not bit address								DPH
82	not bit address								DPL
81	not bit address								SP
80	87	86	85	84	83	82	81	80	P0