



ریزپردازنده

دانشکده کامپیوتر دانشگاه یزد

نیم سال دوم تحصیلی ۹۶-۹۷

ارائه دهنده : پریسا استواری



مجموعه دستورات 8051

انواع دستورات

- دستورات محاسباتی (Arithmetic)
- دستورات منطقی (Logic)
- دستورات انتقال اطلاعات (Data Transfer)
- دستورات بر روی بیت یا متغیرهای بولین (Boolean Variables)
- دستورات کنترل یا انشعاب برنامه (Program Branching)

دستور ANL

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

ANL dest, source

dest = dest AND source •

AND Logic •

- این دستور بر روی یک بایت داده به صورت بیت به بیت انجام می‌شود.
- dest می‌تواند A یا یک خانه‌ی حافظه باشد.

```
MOV  A, #35H    ; A = 35H
ANL  A, #0FH    ; A = A AND 0FH
```

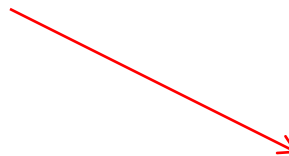
35H	0	0	1	1	0	1	0	1
0FH	0	0	0	0	1	1	1	1
05H	0	0	0	0	0	1	0	1

دستور ANL معمولاً برای صفر کردن بیت‌های خاصی استفاده می‌شود.

دستور ANL

- دستورات ANL، ORL و XOR می‌توانند در یک بایت حافظه داخلی داده (RAM) بدون نیاز به اکومولاتور نیز انجام شوند.

ANL A, 55H	آدرس‌دهی مستقیم
ANL A, @R0	آدرس‌دهی غیرمستقیم
ANL A, R7	آدرس‌دهی ثبات
ANL A, #35H	آدرس‌دهی بلافاصله
<hr/>	
ANL 25H, A	آدرس‌دهی مستقیم
ANL 25H, #44H	آدرس‌دهی مستقیم و بلافاصله



ANL P1, #11110000B
ANL 0F0H, #30H ---> B register

دستور ANL

تعداد سیکل ماشین	تعداد بایت	ترجمه دستور	نوع آدرس دهی	دستور
1	2	01010101 aaaaaaaaa	آدرس دهی مستقیم	ANL A, direct
1	1	0101011i	آدرس دهی غیرمستقیم	ANL A, @Ri
1	1	01011rrr	آدرس دهی ثبات	ANL A, Rn
1	2	01010100 dddddddd	آدرس دهی بلافاصله	ANL A, #data
1	2	01010010 aaaaaaaaa	آدرس دهی مستقیم	ANL direct, A
2	3	01010011 aaaaaaaaa dddddddd	آدرس دهی مستقیم و بلافاصله	ANL direct, #data

دستور ORL

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

ORL dest, source

dest = dest OR source •

OR Logic •

- این دستور بر روی یک بایت داده به صورت بیت به بیت انجام می‌شود.
- dest می‌تواند A یا یک خانه‌ی حافظه باشد.

```
MOV  A, #04H    ; A = 04
ORL  A, #68H    ; A = 6C
```

04H	0	0	0	0	0	1	0	0
68H	0	1	1	0	1	0	0	0
6CH	0	1	1	0	1	1	0	0

دستور ORL معمولاً برای یک کردن بیت‌های خاصی استفاده می‌شود.

دستور ORL

تعداد سیکل ماشین	تعداد بایت	ترجمه دستور	نوع آدرس دهی	دستور
1	2	01000101 aaaaaaaaa	آدرس دهی مستقیم	ORL A, direct
1	1	0100011i	آدرس دهی غیرمستقیم	ORL A, @Ri
1	1	01001rrr	آدرس دهی ثبات	ORL A, Rn
1	2	01000100 dddddddd	آدرس دهی بلافاصله	ORL A, #data
1	2	01000010 aaaaaaaaa	آدرس دهی مستقیم	ORL direct, A
2	3	01000011 aaaaaaaaa ddddddd	آدرس دهی مستقیم و بلافاصله	ORL direct, #data

ORL P1, #11110000B
 ORL 0F0H, #30H ---> B register
 ORL P2, #03H

دستور XRL

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

XRL dest, source

dest = dest XOR source •

XOR Logic •

• این دستور بر روی یک بایت داده به صورت بیت به بیت انجام می‌شود.

• dest می‌تواند A یا یک خانه‌ی حافظه باشد.

• برای معکوس کردن بیت‌ها می‌توان از XRL استفاده کرد. XRL P1, #0FFH

```
MOV  A, #54H
XRL  A, #78H
```

54H	0	1	0	1	0	1	0	0
78H	0	1	1	1	1	0	0	0
2CH	0	0	1	0	1	1	0	0

دستور XRL معمولاً برای معکوس کردن بیت‌های خاصی استفاده می‌شود.

bits of an operand

دستور XRL

تعداد سیکل ماشین	تعداد بایت	ترجمه دستور	نوع آدرس دهی	دستور
1	2	0100010101100101	آدرس دهی مستقیم	XRL A, direct
1	1	0110011i	آدرس دهی غیرمستقیم	XRL A, @Ri
1	1	01101rrr	آدرس دهی ثبات	XRL A, Rn
1	2	01100100 dddddddd	آدرس دهی بلافاصله	XRL A, #data
1	2	01100010 aaaaaaaaa	آدرس دهی مستقیم	XRL direct, A
2	3	01100011 aaaaaaaaa ddddddd	آدرس دهی مستقیم و بلافاصله	XRL direct, #data

XRL P1, #11110000B
 XRL 0F0H, A ---> B register
 XRL 40H, #03H

دستور XRL

- از XOR کردن یک رجیستر یا خانه‌ی حافظه با خودش می‌توان برای پاک کردن آن استفاده کرد. برای مثال دستور `XRL 0E0H, A` مقدار A را صفر می‌کند. (مشابه `CLR A`)

45H	0	1	0	0	0	1	0	1
45H	0	1	0	0	0	1	0	1
00H	0	0	0	0	0	0	0	0

- مثال : پورت P1 را بخوان. اگر مقدار آن برابر با 45H بود، مقدار 99H را به پورت P2 بفرست.

Solution:

```

MOV P2, #00      ;clear P2
MOV P1, #0FFH    ;make P1 an input port
MOV R3, #45H     ;R3=45H
MOV A, P1        ;read P1
XRL A, R3
JNZ EXIT         ;jump if A is not 0
MOV P2, #99H
EXIT: ...

```

XOR برای این استفاده شده که چک شود آیا دو رجیستر مقدار برابر دارند.

Jump Not Zero

EXIT: ...

اگر مقدار دو رجیستر مساوی باشد، در A عدد صفر داریم. دستور JZ و JNZ محتوای A را تست می‌کند.

دستور CPL

CPL A

• اکومولاتور را مکمل می کند.

• Complement register A

```
MOV A, #55H
CPL A           ;now A=AAH
                ;0101 0101 (55H)
                ;becomes 1010 1010 (AAH)
```

• برای داشتن مکمل ۲ کافی است A را با یک جمع کنیم.

• طول دستور یک بایت است و در یک سیکل ماشین اجرا می شود.

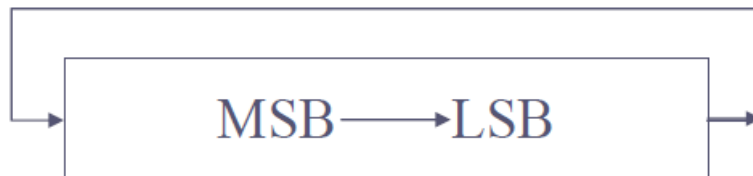
• این دستور بر بیت های پرچم اثری ندارد. (حتی بر بیت توازن زوج. چرا؟)

دستور CLR

CLR A

- اکومولاتور را صفر می کند.
- Clear register A
- طول دستور یک بایت است و در یک سیکل ماشین اجرا می شود.

دستور RR



RR A

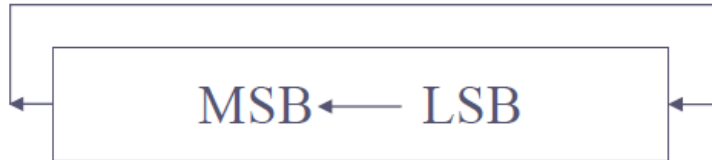
- محتوای A را یک بیت به راست چرخش می‌دهد.
Rotate Right

- محتوای A یک بیت به راست شیفت می‌یابد و کم ارزش‌ترین بیت (LSB) را به پر ارزش‌ترین بیت (MSB) منتقل می‌کند.

MOV A, #36H	; A = 0011 0110
RR A	; A = 0001 1011
RR A	; A = 1000 1101
RR A	; A = 1100 0110
RR A	; A = 0110 0011

- این دستور بر بیت‌های پرچم تاثیری ندارد.
- طول دستور یک بایت است و در یک سیکل ماشین اجرا می‌شود.

دستور RL



RL A

- محتوای A را یک بیت به چپ چرخش می‌دهد.

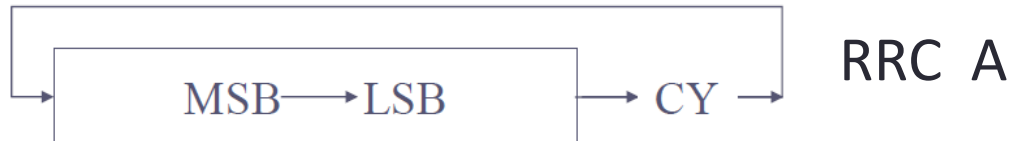
Rotate Left •

- محتوای A یک بیت به چپ شیفت می‌یابد و پر ارزش‌ترین بیت (MSB) را به کم ارزش‌ترین بیت (LSB) منتقل می‌کند.

MOV A, #72H	; A = 0111 0010
RL A	; A = 1110 0100
RL A	; A = 1100 1001

- این دستور بر بیت‌های پرچم تاثیری ندارد.
- طول دستور یک بایت است و در یک سیکل ماشین اجرا می‌شود.

دستور RRC



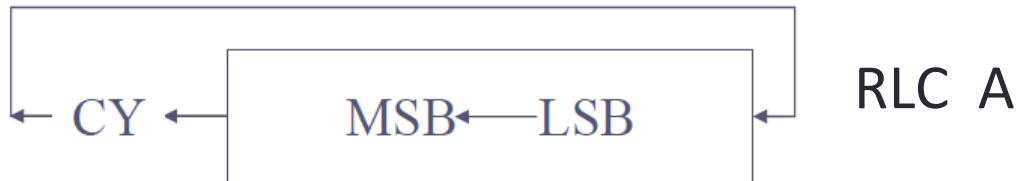
Rotate Right through Carry •

- محتوای A از طریق بیت نقلی C به راست چرخش می کند.
- کم ارزش ترین بیت A به بیت نقلی C منتقل می شود. و بیت نقلی C به پر ارزش ترین بیت A انتقال می یابد. و مابقی بیت های A یک واحد به سمت راست شیفت می خورند.

CLR C	;make CY = 0	
MOV A, #26H	;A = 0010 0110	
RRC A	;A = 0001 0011	CY = 0
RRC A	;A = 0000 1001	CY = 1
RRC A	;A = 1000 0100	CY = 1

- طول دستور یک بایت است و در یک سیکل ماشین اجرا می شود.

دستور RLC



Rotate Left through Carry •

- محتوای A از طریق بیت نقلی C به چپ چرخش می‌کند.
- پر ارزش‌ترین بیت A به بیت نقلی C منتقل می‌شود. و بیت نقلی C به کم ارزش‌ترین بیت A انتقال می‌یابد. و مابقی بیت‌های A یک واحد به سمت چپ شیفت می‌خورند.
- طول دستور یک بایت است و در یک سیکل ماشین اجرا می‌شود.
- مثال : برنامه‌ای بنویسید که تعداد یک‌ها در یک بایت را پیدا کند.

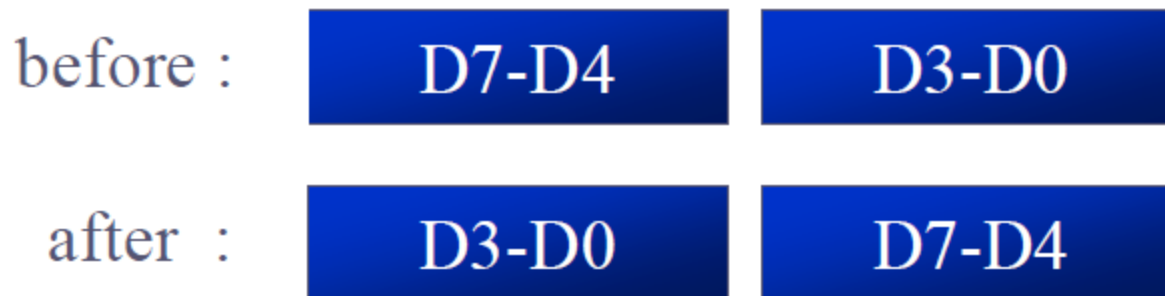
```

MOV     R1, #0
MOV     R7, #8           ;count=08
MOV     A, #97H
AGAIN:  RLC     A
        JNC     NEXT      ;check for CY
        INC     R1        ;if CY=1 add to count
NEXT:   DJNZ    R7, AGAIN
  
```

دستور SWAP

• SWAP A

- این دستور ۴ بیت پر ارزش تر A را با ۴ بیت کم ارزش تر A جابجا می کند.
- این دستور در عملیات با کد BCD مفید خواهد بود.



- دستور SWAP و دستورات چرخشی تنها روی A کار می کنند.
- طول دستور یک بایت است و در یک سیکل ماشین اجرا می شود.

دستور SWAP

• مثال : برنامه‌ای بنویسید که محتوای A را ۳ بیت به چپ بچرخاند.

• روش اول :

RL A

RL A

RL A

• روش دوم :

SWAP A

RR A

- تمام دستورات فوق یک بایتی هستند و در یک سیکل ماشین اجرا می‌شوند.
- پس روش اول ۳ بایت حافظه می‌گیرد و در ۳ سیکل ماشین اجرا می‌شود. اما روش دوم ۲ بایت حافظه می‌گیرد و در ۲ سیکل ماشین اجرا می‌گردد.

دستور SWAP

- مثال : برنامه‌ای بنویسید که بدون استفاده از دستور SWAP کار این دستور را انجام دهد.

(a)

```
MOV    A, #72H    ; A = 72H
SWAP   A           ; A = 27H
```

دستور SWAP

- مثال : برنامه‌ای بنویسید که بدون استفاده از دستور SWAP کار این دستور را انجام دهد.

(a)

```
MOV    A, #72H    ; A = 72H
SWAP   A           ; A = 27H
```

(b)

```
MOV    A, #72H    ; A = 0111 0010
RL     A           ; A = 1110 0100
RL     A           ; A = 1100 1001
RL     A           ; A = 1001 0011
RL     A           ; A = 0010 0111
```



کد اسکی و BCD برای ارقام 0 تا 9

ASCII code and BCD for digits 0 - 9

Key	ASCII (hex)	Binary	BCD (unpacked)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

تبدیل کد BCD فشرده به اسکی

- اگر داده به صورت BCD فشرده (Packed) باشد، برای نمایش بر روی LCD یا ارسال به پرینتر باید به صورت کد اسکی باشد.

Packed BCD		Unpacked BCD		ASCII
29H 0010 1001		02H & 09H 0000 0010 & 0000 1001		32H & 39H 0011 0010 & 0011 1001

مثال

- فرض کنید در A یک عدد BCD فشرده وجود دارد. برنامه‌ای بنویسید که این عدد BCD فشرده را به دو عدد کد اسکی تبدیل کند و آنها را در R2 و R6 بریزد.

```

MOV    A, #29H    ;A=29H, packed BCD
MOV    R2, A      ;keep a copy of BCD data
ANL    A, #0FH    ;mask the upper nibble (A=09)
ORL    A, #30H    ;make it an ASCII, A=39H('9')
MOV    R6, A      ;save it
MOV    A, R2      ;A=29H, get the original
data
ANL    A, #0F0H   ;mask the lower nibble
RR     A          ;rotate right
RR     A          ;rotate right
RR     A          ;rotate right
RR     A          ;rotate right
ORL    A, #30H    ;A=32H, ASCII char. '2'
MOV    R2, A      ;save ASCII char in R2

```

} SWAP A

تبدیل کد اسکی به BCD فشرده

- برای این کار ابتدا کد اسکی را به کد BCD غیرفشرده (Unpacked) تبدیل می‌کنیم (برای خلاص شدن از شر 3)
- سپس دو عدد را ترکیب می‌کنیم را BCD فشرده بدست آید.

key	ASCII	Unpacked BCD	Packed BCD
4	34	0000 0100	0100 0111 or 47H
7	37	0000 0111	

```

MOV    A, #'4'    ;A=34H, hex for '4'
MOV    R1, #'7'   ;R1=37H, hex for '7'
ANL    A, #0FH    ;mask upper nibble (A=04)
ANL    R1, #0FH   ;mask upper nibble (R1=07)
SWAP   A          ;A=40H
ORL    A, R1      ;A=47H, packed BCD

```

مثال

000	'0'	• فرض کنید سه بیت کوچکتر پورت P1 به سه کلید متصل است. برنامه‌ای بنویسید که با توجه به قطع و وصل بودن این کلیدها، کد اسکی متناظر را به پورت P2 بفرستد.
001	'1'	
010	'2'	
011	'3'	
100	'4'	
101	'5'	
110	'6'	
111	'7'	

Solution:

```

MOV     DPTR, #MYTABLE
MOV     A, P1           ;get SW status
ANL     A, #07H         ;mask all but lower 3
MOVC    A, @A+DPTR      ;get data from table
MOV     P2, A           ;display value
SJMP    $               ;stay here

; -----
ORG     400H
MYTABLE DB  '0', '1', '2', '3', '4', '5', '6', '7'
END

```

- برنامه ای بنویسید که یک عدد از پورت یک بخواند، اگر تعداد بیت های یک در چهار بیت پر ارزشتر آن با تعداد بیت های صفر در چهار بیت کم ارزشتر آن برابر بود، عدد یک را به پورت دو بفرستد.

```

MOV P1, #0FFH
L0:  MOV A, P1
      MOV B, A
      MOV R1, #0
      MOV R7, #4
L1:  RLC A
      JNC L2
      INC R1
L2:  DJNZ R7, L1
      MOV A, B
      MOV R2, #0
      MOV R7, #4
L3:  RRC A
      JC L4
      INC R2
L4:  DJNZ R7, L3
      MOV A, R1
      XRL A, R2
      JNZ L5
      MOV P2, #1
L5:  SJMP L0

```

مرجع سریع دستورات منطقی

ANL	A, source
ANL	A, #data
ANL	direct, A
ANL	direct, #data
ORL	A, source
ORL	A, #data
ORL	direct, A
ORL	direct, #data
XRL	A, source
XRL	A, #data

XRL	direct, A
XRL	direct, #data
CLR	A
CPL	A
RL	A
RR	A
RLC	A
RRC	A
SWAP	A

راهنما

Rn	آدرس‌دهی ثبات R0 تا R7
direct	آدرس ۸ بیتی حافظه داده (RAM) داخلی (00H-FFH)
@Ri	آدرس‌دهی غیرمستقیم با استفاده از ثبات‌های R0 یا R1
source	بایت منبع که هر یک از ثبات‌های Rn، آدرس مستقیم (direct)، یا آدرس غیرمستقیم @Ri می‌تواند باشد.
dest	بایت مقصد که هر یک از ثبات‌های Rn، آدرس مستقیم (direct)، یا آدرس غیرمستقیم @Ri می‌تواند باشد.
#data	عدد ۸ بیتی در دستور
#data16	عدد ۱۶ بیتی در دستور
bit	آدرس ۸ بیتی یک بیت اطلاعات
rel	آدرس نسبی یا آفست ۸ بیتی علامت دار
addr11	آدرس ۱۱ بیتی برای صفحه 2k بایتی حافظه
addr16	