معماری کامپیوتر

دانشگاه فنی و حرفه ای قم (واحد دختران)

جلسه ششم

مهندس سراجه حسنی

مروری بر سرفصلهای آنچه گفته شد

1. مباحث پایه مدارهای منطقی
   - معرفی گیت های پایه و منطق کاری آنها
   - مدارهای ترکیبی (کدگشا، کد کننده، مالتی پلکسر)
   - مدارهای ترتیبی (ثبات های شیفت و شمارنده)

2. ساختار حافظه ها
   - کلمه، خط آدرس و خط داده
   - سلسله مراتب حافظه

3. معرفی کامپیوتر پایه
   - ثبات ها و حافظه ی در دسترس برنامه نویس
   - ساختار دستورات
   - معرفی همه ی دستورات کامپیوتر پایه

مثال از برنامه ی ذخیره شده در حافظه

مثال) با توجه به مقادیر موجود در حافظه ی ارائه شده و با فرض اینکه برنامه از خانه ی صفرم حافظه نوشته شده باشد، مقادیر نهایی AC ، E و خانه هایی از حافظه که محتوای آنها تغییر کرده است را مشخص کنید.

Address	Main Memory
000H	7800H
001H	7400H
002H	7100H
003H	100AH
004H	7040H
005H	7040H
006H	7040H
007H	7040H
008H	300AH
009H	7001H
00AH	0050H

مثال از برنامه ی ذخیره شده در حافظه

مثال) با توجه به مقادیر موجود در حافظه ی ارائه شده و با فرض اینکه برنامه از خانه ی صفرم حافظه نوشته شده باشد، مقادیر نهایی AC ، E و خانه هایی از حافظه که محتوای آنها تغییر کرده است را مشخص کنید.

Address	Main Memory		Instruction		Result
000H	7800H	→	CLA	→	AC = 0
001H	7400H	→	CLE	→	E = 0
002H	7100H	→	CME	→	E = 1
003H	100AH	→	ADD [00AH]	→	AC = AC + [00AH] => AC = 0050H
004H	7040H	→	CIL	→	Rotate AC & E to left => AC = 00A1H , E = 0
005H	7040H	→	CIL	→	Rotate AC & E to left => AC = 0142H , E = 0
006H	7040H	→	CIL	→	Rotate AC & E to left => AC = 0284H , E = 0
007H	7040H	→	CIL	→	Rotate AC & E to left => AC = 0508H , E = 0
008H	300AH	→	STA [00AH]	→	[00AH] = AC => [00AH] = 1005H
009H	7001H	→	Halt		
00AH	~~0050H~~ **1005H**				

AC = 1005H
E = 0
[00AH] = 1005H

فرم کلی برنامه نویسی کامپیوتر پایه

- معمولا محتوای حافظه برای ارائه ی برنامه ها بکار گرفته نمیشود.
- فرم کلی برنامه نویسی زیر برای سهولت در نمایش برنامه ها در نظر گرفته شده است:

برچسب,　　دستور　　پارامتر　　I

- ✓ "برچسب ها" به جای آدرس خانه های حافظه استفاده میشوند.
- ✓ دستور همان نام دستورات است. مانند CLE، ADD ، HLT و ...
- ✓ در صورتی که دستور از نوع حافظه ای باشد، پارامتر مقدار مربوط به آن دستور را خواهد گرفت.
- ✓ اگر آدرس دهی در دستور حافظه ای از نوع غیر مستقیم باشد، I برای آن دستور ذکر میشود. در غیر این صورت یعنی در حالتی که آدرس دهی پارامتر به صورت مستقیم است، I گذاشته نمیشود.
- ✓ برنامه ها با شبه دستور ORG شروع میشوند که پارامتر روبروی این شبه دستور نشان میدهد برنامه از کدام خانه ی حافظه نوشته شده است.
- ✓ برنامه ها با شبه دستور END پایان میابند. توجه کنید دستور HLT، در انتهای دستورات می آید و کامپیوتر را خاموش میکند در حالی که ممکن است عملوندهایی در انتهای برنامه تعریف شده باشد که قاعدتا جز برنامه هستند اما دستور حساب نمیشوند. شبه دستور END در انتهای کل برنامه گذاشته میشود.

فرم کلی برنامه نویسی کامپیوتر پایه (ادامه ...)

مثال) مثال اسلاید شماره ۲ را با فرم جدید بازنویسی کنید.

Address	Main Memory
000H	7800H
001H	7400H
002H	7100H
003H	100AH
004H	7040H
005H	7040H
006H	7040H
007H	7040H
008H	300AH
009H	7001H
00AH	0050H

→

?

فرم کلی برنامه نویسی کامپیوتر پایه (ادامه ...)

مثال) مثال اسلاید شماره ۲ را با فرم جدید بازنویسی کنید.

Address	Main Memory
000H	7800H
001H	7400H
002H	7100H
003H	100AH
004H	7040H
005H	7040H
006H	7040H
007H	7040H
008H	300AH
009H	7001H
00AH	0050H

→

```
     ORG   000H
     CLA
     CLE
     CME
     ADD   L1
     CIL
     CIL
     CIL
     CIL
     STA   L1
     HLT
L1,  Hex   0050
     END
```

فرم کلی برنامه نویسی کامپیوتر پایه (ادامه ...)

مثال) برنامه ای بنویسید که دو خانه از حافظه به آدرس های A و B با مقادیر دلخواه را از هم کم کرده و نتیجه را در خانه ی A ذخیره کند. این برنامه از خانه ی صفرم حافظه شروع شود.

$$A = A - B$$

فرم کلی برنامه نویسی کامپیوتر پایه (ادامه ...)

مثال) برنامه ای بنویسید که دو خانه از حافظه به آدرس های A و B با مقادیر دلخواه را از هم کم کرده و نتیجه را در خانه ی A ذخیره کند. این برنامه از خانه ی صفرم حافظه شروع شود.

$$A = A - B \longrightarrow A = A + (-B) \longrightarrow A = A + 2\text{'s Complement}(B)$$

ORG 000H	برنامه از خانه ی صفرم حافظه شروع شود.
LDA B	B را در ثبات انباشته گر بارگذاری کن.
CMA	ثبات انباشته گر را مکمل کن.
INC	ثبات انباشته گر را با یک جمع کن. (مکمل دو گرفتن در این مرحله تمام میشود)
ADD A	A را با ثبات انباشته گر جمع کن. (حاصل جمع در خانه ی AC ذخیره شده است.)
STA A	ثبات انباشته گر را در خانه ی A ذخیره کن.
HLT	کامپیوتر را خاموش کن.
A, DEC 50	
B, DEC 10	
END	

پیاده سازی حلقه

- در برنامه ها به جز محاسبات معمولی، حلقه (loop) هم وجود دارد. بعضی از حلقه ها شمارنده دارند و برخی دیگر فقط با بررسی یک شرط تمام میشوند.

مثال) برنامه ی زیر را بررسی کرده و مشخص کنید چه کاری انجام میدهد؟

```
      ORG   100H
      LDA   ADS
      STA   PTR
      LDA   NBR
      STA   CTR
      CLA
Lop,  ADD   PTR   I
      ISZ   PTR
      ISZ   CTR
      BUN   Lop
      STA   Sum
      HLT
```

```
ADS,  HEX   0150
PTR,  HEX   0000
NBR,  DEC   -100
CTR,  HEX   0000
Sum,  HEX   0000
      ORG   150H
      DEC   100
      DEC   99
         .
         .
         .
      DEC   0
      END
```

پیاده سازی حلقه (ادامه ...)

مثال) برنامه ی زیر را بررسی کرده و مشخص کنید چه کاری انجام میدهد؟

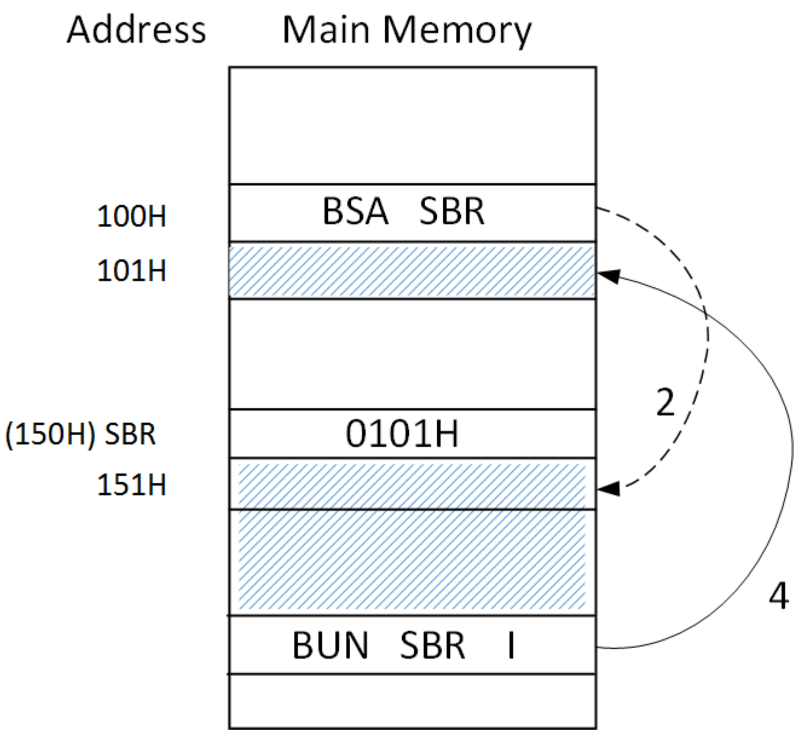
1.	ORG 100H	1.	برنامه از خانه ی ۲۵۶ حافظه نوشته شده است
2.	LDA ADS	2.	AC = 0150H
3.	STA PTR	3.	[PTR]= 0150H
4.	LDA NBR	4.	AC = -100
5.	STA CTR	5.	[CTR] = -100
6.	CLA	6.	AC = 0
7.	Lop, ADD PTR I	7.	Lop, AC = AC + [[PTR]]
8.	ISZ PTR	8.	[PTR] = [PTR] + 1; If ([PTR] == 0) skip next inst.
9.	ISZ CTR	9.	[CTR] = [CTR] + 1; If ([CTR] == 0) skip next inst.
10.	BUN Lop	10.	Jump to Lop
11.	STA Sum	11.	[Sum] = AC
12.	HLT	12.	Halt the computer
13.	ADS, HEX 0150	13.	[ADS] = 0150H
14.	PTR, HEX ~~0000~~ 0150H	14.	[PTR] = 0150H
15.	NBR, DEC -100	15.	[NBR] = -100
16.	CTR, ~~HEX 0000~~ -100	16.	[CTR] = -100
17.	Sum, ~~HEX 0000~~ 5050	17.	[Sum] = 5050
18.	ORG 150H	18.	خط بعد از خانه ی ۱۵۰H به بعد نوشته شود
19.	DEC 100	19.	[150H] = 100
20.	DEC 99	20.	[151H] = 99
	.		.
	.		.
	.		.
21.	DEC 1	21.	[1B4H] = 1
22.	END	22.	انتهای خطوط برنامه

این برنامه محتوای ۱۰۰ خانه پشت سرهم از حافظه را که آدرس آنها از 150H شروع میشود، با هم جمع کرده و حاصل را در خانه ای از حافظه که آدرس آن Sum است، ذخیره میکند.

صدا زدن زیر برنامه (SUBROUTINE)

- معمولا دسته ای از دستورات که اجرای آنها در برنامه بارها تکرار میشود را داخل تابع یا زیر برنامه نوشته و در میانه ی دستورات اصلی برنامه، هر کجا نیاز بود، به آن زیر برنامه رجوع میکنند.

برای نوشتن زیر برنامه و رجوع به آن، ساختار برنامه اصلی و زیر برنامه باید به صورت زیر باشد:



۱- هر جا نیاز بود به زیر برنامه با دستور BSA میپریم.
۲- آدرس بازگشت در اولین خانه ی زیر برنامه ذخیره میشود. این کار به صورت اتوماتیک با اجرای دستور BSA انجام میشود. به این ترتیب از آنجایی که آدرس بازگشت در اولین خط از زیربرنامه نوشته میشود، این خانه نباید حاوی هیچ دستوری باشد.
۳- دستورات زیر برنامه اجرا میشوند.
۴- آخرین دستور زیر برنامه باید کار بازگشت به برنامه ی اصلی را انجام دهد. از آنجایی که آدرس بازگشت در اولین خط زیربرنامه ذخیره شده است، پس با پرش بدون شرط به صورت غیر مستقیم، میتوان به برنامه ی اصلی بازگشت.

صدا زدن زیر برنامه (SUBROUTINE)

مثال) برنامه ای بنویسید که دو خانه از حافظه به آدرس های X و Y و با مقادیر دلخواه را به صورت جداگانه چهار مرتبه به سمت راست دوران داده و سپس چهار بیت پایین آنها را صفر کرده و در همان خانه های X و Y ذخیره کند. دوران ها و صفر کردن چهار بیت پایین را در زیر برنامه بنویسید. برنامه از خانه ی صفرم حافظه شروع شود.

صدا زدن زیر برنامه (SUBROUTINE)

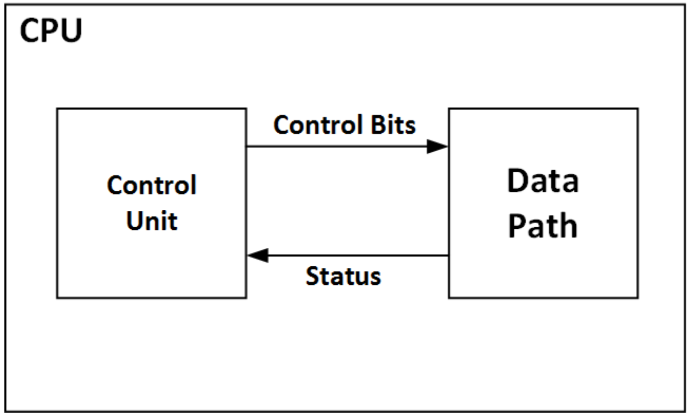
مثال) برنامه ای بنویسید که دو خانه از حافظه به آدرس های X و Y و با مقادیر دلخواه را به صورت جداگانه چهار مرتبه به سمت راست دوران داده و سپس چهار بیت پایین آنها را صفر کرده و در همان خانه های X و Y ذخیره کند. دوران ها و صفر کردن چهار بیت پایین را در زیر برنامه بنویسید. برنامه از خانه ی صفرم حافظه شروع شود.

```
        ORG   000H
        LDA   X
        BSA   SBR1
        STA   X
        LDA   Y
        BSA   SBR1
        STA   Y
        HLT
SBR1,   DEC   0
        CLE
        CIR
        CIR
        CIR
        CIR
        AND   Msk
        BUN   SBR1   I
```

```
X,    HEX   1234H
Y,    HEX   4321H
Msk,  HEX   FFF0H
      END
```

طراحی کامپیوتر پایه

- تا اینجا با مجموعه ثبات ها و حافظه ی در دسترس برنامه نویس و دستورالعملهای کامپیوتر پایه آشنا شدیم. برای ساخت این کامپیوتر (منظور همان CPU کامپیوتر است)، آن را به اجزای کوچکتری میشکنیم:



1. Data Path (مسیر داده) : تمام قسمت هایی که محاسبات انجام میدهند و اجرای دستورات در آنها صورت میگیرد در این قسمت قرار دارد. (واحد محاسبات و منطق، ثبات ها، حافظه و ...)
2. Control Unit (واحد کنترل) : ترتیب انجام عملیاتی که باید در واحد مسیرداده اجرا شوند و روند موازی بودنشان، در این واحد مشخص میگردد.

✓ دو واحد مسیر داده و کنترل، دائما با هم در ارتباط هستند. به این صورت که واحد مسیر داده وضعیت خود را در قالب مجموعه ای از بیت ها به واحد کنترل میدهد و واحد کنترل نیز بر اساس آنها، مجموعه از بیت های کنترلی تولید کرده و روند کار واحد مسیر داده را با استفاده از آنها کنترل میکند.

طراحی کامپیوتر پایه (ادامه ...)

- برای طراحی واحد های مسیرداده و کنترل، باید همه ی دستورها تک به تک بررسی شوند و بر اساس کاری که هر دستور انجام میدهد و منابعی که برای انجام این کارها نیاز است، قسمت های مختلف این دو واحد و نحوه ی ارتباط بین این قسمت ها مشخص شود.
- برای مثال در واحد مسیرداده قطعا علاوه بر ثبات های در اختیار برنامه نویس که بسیار محدود نیز هستند، ثبات های دیگری نیاز است. علاوه بر این، نوع ارتباط این ثبات ها و واحد حافظه نیز باید به صورت کامل در نظر گرفته شود.
- میتوان همین روند توضیح با زبان انسان را ادامه داد و به یک طراحی از واحد مسیرداده رسید. اما به علت ابهام موجود در زبان انسان و پیچیدگی دستورات کامپیوتر پایه، زبانی اختصاصی و علمی برای این کار (توصیف سخت افزاری دستورات اسمبلی) ابداع شده است. به این ترتیب میتوان با به کارگیری این زبان، سخت افزار معادل هر دستور را توصیف و در نهایت واحد های مسیرداده و کنترل را طراحی کرد.
- به این زبان «زبان انتقال ثبات» یا (RTL)"Register Transfer Language" میگویند.