فصل ۱

معرفی سخت افزار سیمپلوس

۱.۱ مقدمه

سیمپلوس[1]، یک کامپیوتر فرضی است که ساختار آن معرف بسیاری از کامپیوترهای واقعی امروزی است. گرچه تا کنون نمونهٔ سخت افزاری آن ساخته نشده‌است اما ما در این درس می‌خواهیم نمونهٔ شبیه‌سازی شده آنرا طراحی نماییم. با توجه به اینکه خصوصیات کاری و رفتاری این ماشین کاملاً قابل توصیف است، لذا شبیه‌سازی آن کار سختی نمی‌باشد. همین خصوصیات ویژگی توسعه‌پذیر بودن آنرا نمایان می‌کند. این مستند بر آن است تا خصوصیات پایه‌ای این ماشین مجازی را بصورت کامل توصیف نماید. برای ساده‌تر شدن توسعه سیمپلوس در ابتدا تنها دستورالعمل‌های پایه‌ای ماشین بررسی خواهند شد. در ادامه برای امکان اجرای برنامه‌های قوی‌تر دستور العمل‌های پیچیده‌تر به ماشین اضافه خواهد شد. برای اینکار ما ساختار سخت‌افزاری ماشین را تغییر می‌دهیم تا امکان انجام کارهای پیچیده‌تر وجود داشته باشد.

۲.۱ اجزای سخت‌افزاری سیمپلوس

ماشین سیمپلوس دارای یک واحد کنترل مرکزی (CPU)، یک حافظه ۱۰۰ کلمه‌ای، صفحه کلید به عنوان دستگاه ورودی و صفحه نمایش به عنوان خروجی می‌باشد.

CPU : قلب سیمپلوس می‌باشد. cpu می‌تواند مجموعه‌ای محدود از عملیات ساده را انجام دهد، هر بار یک دستور. این بخش با استفاده از مراحل زیر هر عملیاتی را انجام می‌دهد:

۱. عددی که در ثبات شمارنده دستور (PC) هست را می‌خواند،
۲. به حافظه‌ای با شماره خوانده شده مراجعه کرده و دستورالعمل قرار داده شده در آن را در ثبات دستورالعمل (IS) کپی می‌کند،
۳. یکی به مقدار ثبات شمارنده دستور اضافه می‌کند،
۴. دستورالعمل موجود در ثبات IS را اجرا می‌کند،
۵. به مرحله یک برمی‌گردد.

cpu شامل چهار ثبات می‌باشد:

۱. PC: ثبات شمارنده برنامه[2] شماره دستوری که در مرحله بعد اجرا می‌شود را نگهداری می‌نماید.
۲. IS: ثبات دستورالعمل[3] وظیفه نگهداری دستورالعمل جاری را برعهده دارد.
۳. ACC: ثبات انباره[4] وظیفه نگهداری مقدار جاری را بر عهده دارد.
۴. FLG: ثبات پرچم[5] وضعیت جاری سیستم را نگهداری می‌نماید.

حافظه: برای این کامپیوتر ۱۰۰ کلمه[6] حافظه در نظر گرفته شده است. هر کلمه معادل یک عدد صحیح می‌باشد. حافظه مربوط به برنامه و داده‌ها در یکجا تعبیه شده است. یعنی حافظه داده‌ها بلافاصله بعد از فضای برنامه شروع می‌شود.

[1] Simplus
[2] Program Counter
[3] InStruction
[4] ACCumulator
[5] FLaGs
[6] word

۳.۱ دستورالعمل‌های ماشین

همه دستورالعمل‌های ماشین سیمپلوس عدد صحیح بوده و به یکی از صورت‌های زیر استفاده می‌شوند:

نوع اول: ۵ــ بیت کد دستورالعمل، ۱۱ــ بیت پرکننده[۷]

نوع دوم: ۵ــ بیت کد دستورالعمل، ۱۱ــ بیت آدرس.

که در آن آدرس عدد صحیح ۱۱ــ بیتی و بدون علامت است. دستورالعمل‌های ماشین سیمپلوس به همراه معنای هریک از آنها در جدول ۱.۳ آمده است. زبان اسمبلی یک زبان برنامه‌نویسی سطح پایین است که به جای کدهای دستورالعمل عددی از حفظیات و برچسب‌ها استفاده می‌کند.

جدول ۱.۱: لیست دستورالعمل‌های ماشین سیمپلوس

کد عددی	کد حفظی	دستورالعمل	توضیحات
01xx	ADD	ADD	مقدار ذخیره شده در حافظه xx را به مقدار موجود در انباره اضافه می‌کند. نکته: محتوای حافظه تغییر نمی‌کند، و عمل‌های انباره برای دستورالعمل‌های جمعی که جمع‌های بیشتر از ۳ را انجام می‌دهند، تعریف نشده‌اند. به‌طور مشابه برای تفریق، می‌توان یک پرچم منفی بر روی سرریز در نظر گرفت.
02xx	SUB	SUBTRACT	مقدار ذخیره شده در حافظه xx را از مقدار موجود در انباره تفریق می‌کند. نکته: محتوای حافظه تغییر نمی‌کند.
03xx	MUL	MULTIPLY	مقدار ذخیره شده در حافظه xx را در مقدار موجود در انباره ضرب می‌کند. نکته: محتوای حافظه تغییر نمی‌کند.
04xx	DIV	DIVISION	مقدار ذخیره شده در حافظه xx را بر مقدار موجود در انباره تقسیم می‌کند. نکته: درصورت بروز تقسیم بر صفر FLG با مقدار ۱ پر می‌شود.
05xx	STA	STORE	ذخیره مقادیر موجود در انباره در حافظه xx (مخرب). نکته: مقادیر انباره بدون تغییر باقی می‌مانند (تخریب نمی‌شوند)، اما مقادیر حافظه یا مقداری که در آنجا بوده جایگزین می‌شوند (تخریب می‌شوند).
06xx	LDA	LOAD	مقدار حافظه xx را بارگذاری می‌کند (از بین نمی‌برد) و آن را به انباره وارد می‌کند (از بین می‌برد).
07xx	BRA	BRANCH ALWAYS	شمارنده برنامه را به آدرس داده شده تنظیم می‌کند (مقدار xx). یعنی مقدار xx دستورالعمل بعدی خواهد بود که اجرا می‌شود.
08xx	BRZ	BRANCH IF ZERO	اگر مقدار انباره برابر با صفر بود، شمارنده برنامه را برابر مقدار xx قرار بده. در غیر این صورت هیچ عملی انجام نده. نکته: تا زمانی که برنامه در حافظه ذخیره می‌شود، تمامی داده‌ها و دستورالعمل‌های برنامه فرمت آدرس/مکان یکسانی دارند.
09xx	BRP	BRANCH IF POSITIVE	اگر مقدار انباره مثبت باشد، شمارنده برنامه بر روی xx تنظیم می‌شود. در غیر اینصورت هیچ عملی انجام نمی‌دهد.
10xx	BRN	BRANCH IF NEGATIVE	اگر مقدار انباره منفی باشد، شمارنده برنامه بر روی xx تنظیم می‌شود. در غیر اینصورت هیچ عملی انجام نمی‌دهد.
11xx	BRG	BRANCH IF FLAGS	اگر مقدار ثبات پرچم یک باشد، شمارنده برنامه بر روی xx تنظیم می‌شود. در غیر اینصورت هیچ عملی انجام نمی‌دهد.
1201	INP	INPUT	برو به صندوق ورودی، مقدار را از کاربر دریافت کن، و در انباره قرار بده. نکته: این عمل باعث بازنویسی هر مقداری که داخل انباره بوده‌است می‌شود (مخرب).
1202	OUT	OUTPUT	مقدار را از انباره به صندوق خروجی کپی کن. نکته: محتوای انباره تغییر نمی‌کند (غیر مخرب).
0000	HLT	HALT	برنامه را پایان ده
	DAT	DATA	این یک دستور اسمبلی است که به سادگی مقدار را درون حافظهٔ دردسترس بعدی بار می‌کند. همچنین می‌تواند در تلفیق با برچسب‌ها برای تعریف متغیرها استفاده شود.

اگرچه سیمپلوس تنها از یک دستهٔ محدود از حفظیات استفاده می‌کند، اما راحتی استفاده از یک حفظی برای هر دستورالعمل از زبان اسمبلی ــ که در زیر نشان داده شده ــ سادگی کار را روشن می‌سازد.

[۷] پر کننده مقداری بی‌معنی است که تنها برای پرشدن محل مورد نظر استفاده می‌شود. بیشتر اوقات از صفر یا فضای‌خالی به‌عنوان پرکننده استفاده می‌شود

در زیر چند مثال از برنامه به زبان اسمبلی سیمپلس آمده است:
برنامه زیر قرار است دو عدد را از کاربر دریافت کرده و حاصل تفریق آنها را در خروجی نمایش دهد:

```
        INP
        STA FIRST
        INP
        STA SECOND
        LDA FIRST
        SUB SECOND
        OUT
        HLT
FIRST   DAT
SECOND  DAT
```

خروجی:

INPUT	OUTPUT
5	-2
7	

مثال زیر عدد مثبتی را از کاربر دریافت کرده و از آن عدد تا صفر را در خروجی نمایش می‌دهد:

```
      INP
      OUT      // Initialize output
LOOP  BRZ QUIT // If the accumulator value is 0, jump to the memory address labeled QUIT
      SUB ONE  // Label this memory address as LOOP, The instruction will then
               // subtract the value stored at address ONE from the accumulator
      OUT
      BRA LOOP // Jump (unconditionally) to the memory address labeled LOOP
QUIT  HLT      // Label this memory address as QUIT
ONE   DAT 1    // Store the value 1 in this memory address,
               // and label it ONE (variable declaration)
```

خروجی:

INPUT	OUTPUT
5	5
	4
	3
	2
	1
	0

برنامه زیر دو مقدار صحیح را از کاربر دریافت کرده . مقدار بیشینهٔ آنها را در خروجی نمایش می‌دهد:

```
      INP
      STA num1
      INP
      STA num2
      SUB num1
      BRP pos
      LDA num1
      OUT
      BRA exit
pos   LDA num2
      OUT
```

```
exit HLT
num1 DAT
num2 DAT
```

خروجی:

INPUT	OUTPUT
5	7
7	

برنامه زیر ده عدد اول مثلثی را در خروجی نمایش می‌دهد:

```
loop    LDA number
        ADD counter
        OUT
        STA number
        LDA counter
        ADD one
        STA counter
        LDA ten
        SUB counter
        BRP loop
        HLT
counter DAT 1
number  DAT 0
one     DAT 1
ten     DAT 10
```

خروجی:

INPUT	OUTPUT
	1
	3
	6
	10
	15
	21
	28
	36
	45
	55

برنامه زیر عددی را از کاربر دریافت کرده سپس مکعب آنرا را در خروجی نمایش می‌دهد و این کار را تا زمانی که کاربر عدد صفر وارد نکرده ادامه می‌دهد:

```
START  LDA ONE     // Initialize for multiple program run
       STA RESULT
       STA COUNT
       INP          // User provided input
       BRZ END      // Branch to program END if input = 0
       STA VALUE    // Store input as VALUE
LOOP   LDA RESULT   // Load the RESULT
       MUL VALUE    // multiply VALUE by RESULT,
       STA RESULT   // Store the new RESULT
       LDA COUNT    // Load the COUNT
       ADD ONE      // Add ONE to the COUNT
       STA COUNT    // Store the new COUNT
```

```
        SUB TIMES    // Subtract COUNT from TIMES
        BRZ ENDLOOP  // If zero (VALUE has been multiply to RESULT by 3 times),
                     // branch to ENDLOOP
        BRA LOOP     // Branch to LOOP to continue multiplying VALUE to RESULT
ENDLOOP LDA RESULT    // Load RESULT
        OUT          // Output RESULT
        BRA START    // Branch to the START to initialize and get another input VALUE
END      HLT          // HALT - a zero was entered so done!
RESULT   DAT          // Computed result (defaults to 0)
COUNT    DAT         // Counter (defaults to 0)
TIMES    DAT 3       // limits
ONE      DAT 1        // Constant, value of 1
VALUE    DAT          // User provided input, the value to be squared (defaults to 0)
ZERO     DAT          // Constant, value of 0 (defaults to 0)
```

خروجی:

INPUT	OUTPUT
4	64
2	8
3	27
0	