فصل سوم

اصول برنامه‌نویسی به زبان اسمبلی

مقدمه

همان‌طور که قبلاً گفته شد، نوشتن و درک برنامه به زبان ماشین کار مشکل و خسته‌کننده‌ی است لذا امروزه ترجیح می‌دهند کمتر با زبان ماشین کار کرده و برنامه‌های کامپیوتر را به زبان اسمبلی بنویسند. زبان اسمبلی به جای استفاده از اعداد، معمولاً از حروف الفبای انگلیسی استفاده می‌کند و برنامه‌ی مترجم یا اسمبلر، برنامه به زبان اسمبلی را به زبان ماشین ترجمه می-نماید. گرچه اصول زبان اسمبلی برای همه کامپیوترها یکسان است ولی هر ماشین دارای زبان اسمبلی ویژه خود می‌باشد بنابراین زبان اسمبلی همانند زبان ماشین وابسته به ساختمان ماشین است و دستورهای زبان اسمبلی در انواع ماشین‌ها متفاوت می‌باشد.

هر دستور زبان اسمبلی فقط به یک دستور زبان ماشین ترجمه می‌شود. بنابراین اگر برنامه‌ایی در زبان اسمبلی N دستور داشته باشد، این برنامه توسط مترجم اسمبلر تبدیل به N دستور زبان ماشین خواهد شد. در صورتی که این موضوع در برنامه‌های به زبان دیگر که دارای کامپایلر هستند صادق نبوده و ترجمه هر دستور برنامه‌ها ممکن است تولید چند دستور به زبان ماشین بنماید. استفاده از زبان اسمبلی (به جای زبان ماشین) کار برنامه‌نویسی را ساده می‌نماید، چون یادگیری سمبول‌های اسمبلی بسیار آسانتر از کدهای باینری، یا هگزادسیمال زبان ماشین است.

به کمک زبان اسمبلی می‌توان به تمام عملیات ماشین تسلط داشت و کلیه موقعیت‌های پیش آمده را بررسی نمود و به محتوای ثبات‌های پروسسور دسترسی پیدا کرد در حالی که در زبان‌های برنامه‌نویسی سطح بالا امکان استفاده از این تسهیلات نیست. لذا در کاربردهایی که نیاز به دسترسی مستقیم به سخت‌افزار می‌باشد، تهیه و اجرای برنامه‌ها به زبان اسمبلی، نسبت به برنامه‌های سطح بالا بسیار سریع‌تر می‌باشد به همین دلیل است که، اکثر برنامه‌های سیستم عامل و همچنین برنامه‌های کامپیوتری دستگاه‌های صنعتی و میکروکنترلرها به زبان اسمبلی می‌باشند.

نمونه‌ای از برنامه اسمبلی

همان‌طور که ملاحظه گردید، برنامه‌نویسی به زبان اسمبلی بسیار آسان‌تر از برنامه‌نویسی به زبان ماشین می‌باشد. اصولاً هر برنامه به زبان اسمبلی از تعدادی دستورات زبان اسمبلی تشکیل می‌گردد. به عنوان مثال برنامه زیر نمونه‌ایی از یک برنامه اسمبلی می‌باشد که از یک سری دستورات زبان اسمبلی تشکیل شده است. این برنامه محتویات دو خانه حافظه به آدرس‌های DATA1 و DATA2 که در کد اسکی هستند را با هم جمع می‌نماید و نتیجه جمع در (کد باینری) خانه DATA3 قرار می‌دهد و در پایان به سیستم عامل برمی‌گردد.

```
page 110,100
title 'Exam1.asm'
stacksg segment stack  'stack'
        dw 32h DUP(0)
stacksg ends
datasg segment 'data'
        Data1 db 10
        Data2 db 20
        Data3 db ?
datasg ends
```

```
codesg segment 'code'
        assume ss:stacksg,ds:datasg,cs:codesg
main proc far
        mov ax,datasg
        mov ds,ax
        mov al,Data1
        mov bl,Data2
        add al,bl
        mov Data3,al
exit:
        mov ax,4c00h
        int 21h
main    endp
codesg  ends
end main
```

قالب یا فرمت دستورات زبان اسمبلی

قالب دستورات زبان اسمبلی به صورت کلی دارای قسمت یا فیلدهای برچسب دستور، کد اجرا، عملوند یا اپراندها و توضیحات می‌باشد.

Label	:	Opcode	Operand1	,	Operand2	, . . . ;	Comment
برچسب	:	کد اجرا	اُپراند۱ یا عملوند۱	,	اُپراند ۲ یا عملوند۲	, . . . ;	توضیحات

به عنوان مثال: MAIN: MOV AH,0 ; Move zero to

در دستور فوق، بر چسب دستور، MAIN است. کد اجرا سمبول MOV می‌باشد که معنی آن این است که صفر به عملوند یا اپراند AH انتقال می‌یابد و بعد از نقطه‌ویرگول توضیح دستور به زبان انگلیسی نوشته شده است.

برچسب

برچسب نام اختیاری است که به دستور یا اطلاعات داده می‌شود. قسمت یا فیلد برچسب، به هشت حرف محدود می‌شود و در اولین قسمت دستور اسمبلی قرار می‌گیرد و بعد از آن دو نقطه می‌باشد که علامت برچسب دستور است. اصولاً برچسب دستور اختیاری است و برای مراجعه به آن دستور در دستورالعمل‌های پرش به کار برده می‌شود.

کد اجرا یا عملگر

کد اجرای دستور، نوع کاری را که دستور باید اجرا نماید، مشخص می‌کند و معمولاً به صورت سمبولیک در زبان انگلیسی نوشته می‌شود مانند ADD، MOV، SUB.... که به ترتیب نمایش جمع، انتقال و تفریق و .... می‌باشند و اصولاً کد اجرا سه یا چهار حرفی می‌باشد.

عملوند یا اپراند

اپراند می‌تواند آدرس حافظه یا ثبات و یا عدد ثابت باشد. تعدادی از دستورات ممکن است یک یا دو یا چند اپرند داشته باشند. در بعضی دیگر مانند MOVSD یا NOP ممکن است اصلاً اپراند نداشته باشند.

توضیحات برنامه

برای توضیح و شرح کار دستورها و برنامه می‌توان با استفاده از علامت (نقطه ویرگول) عبارتی را به عنوان توضیح در هر دستور یا خط برنامه نوشت البته این قسمت در برنامه اسمبلی اختیاری است. به عنوان مثال:

STRAT: JMP MAIN ; Jump over data

که در دستور فوق START برچسب دستور، JMP کد اجرا و MAIN اپراند یا آدرس حافظه و بالاخره Jump over data توضیح این دستور است که می‌گوید از قسمت داده‌ها پرش کند.

روش نام‌گذاری آدرس اطلاعات و برچسب دستور

همان‌طور که در فصل اول ملاحظه شد خانه‌های حافظه با آدرس مشخص می‌شوند. به عنوان مثال 1500 آدرس یک خانه حافظه است و اگر بخواهیم محتوای این خانه از حافظه را به ثباتی مانند R0 منتقل کنیم از دستور:

MOV R0,[1500]

استفاده می‌کنیم. در حقیقت [1500] یعنی محتوای خانه حافظه‌ای که آدرس آن 1500 است. ولی اگر در خانه حافظه 1500 نتیجه‌ی جمع چند عدد قرار گرفته باشد، بهتر است به جای به کار بردن آدرس 1500، نام سمبولیک اختیاری مثلاً SUM را به این آدرس اختصاص دهیم. در این صورت دستور بالا به صورت:

MOV R0,SUM

نوشته می‌شود. لذا کسی که این دستور را می‌خواند، متوجه می‌شود که مجموع چند عدد در آدرس SUM به ثبات R0 منتقل شده است. بنابراین در زبان اسمبلی معمولاً به جای استفاده از آدرس عددی، نام سمبولیک اختیاری به آدرس اطلاعات حافظه می‌دهیم که در ارتباط با مفهوم مسئله باشد. البته نام آدرس اطلاعات کاملاً اختیاری است. علاوه بر این، برای مراجعه به دستورات، به آدرس دستور در حافظه نیز، نام سمبولیکی اختصاص می‌دهیم که برچسب دستور نامیده می‌شود، مانند دستور:

LOOP1: MOV R0,10

که LOOP بر چسب یا افست آدرس دستور در حافظه است و نام سمبولیکی است که ما به آدرس آن دستور داده‌ایم. البته هر نامی برای آدرس اطلاعات یا آدرس دستور که با حروف A تا Z (حروف کوچک یا بزرگ) و رقم‌های 0 تا 9 باشد، می‌توان انتخاب نمود بشرطی که اولین حرف یک رقم نباشد و جزء کلمات رزرو شده مانند نام ثبات AX، BXو CX ... نباشد. معمولاً نام‌های مذکور را حداکثر 7 حرفی مانند COUNT، TOTAL1، JAMEH2، MAIN، LOOP2، ADDRES1،.... انتخاب می‌کنند.