مقدمه

زبان اسمبلی قدیمی ترین زبان برنامه نویسی سطح پایین بعد از زبان ماشین است که ساختار و عملکردی وابسته به ماشین دارد و وسیله خوبی برای یادگیری نحوه کار کامپیوتر، سیستم عامل، کامپایلرها و زبان های سطح بالا است.

مقایسه زبان اسمبلی و زبان های سطح بالا
زبان ماشین
زبان اسمبلی چیست؟
اسمبلر
هدف از یادگیری زبان اسمبلی

مقایسه زبان اسمبلی و زبان های سطح بالا

دو دسته اصلی زبان های برنامه نویسی عبارتند از :

۱. زبان های سطح بالا
- مانند C++ ، Pascal، Java و Visual Basic.
۲. زبان های سطح پایین
- زبان ماشین
- زبان اسمبلی

اکثر برنامه نویسان در لایه زبان سطح بالا کار می کنند که هر عبارت آن به چند دستورالعمل ماشین ترجمه می شود. برنامه های نوشته شده در زبان های سطح بالا خصوصا زبان های شی گرا راحت تر، سریع تر و با هزینه کمتر پیاده سازی و نصب می شوند .

زبان اسمبلی یک زبان سطح پایین است و اغلب هنگام ارتباط با سیستم عامل، دسترسی مستقیم به خواص کلیدی ماشین یا برای بهینه کردن قسمت های حساس برنامه های کاربردی و افزایش سرعت اجرای آنها استفاده می شود. برنامه نویسی زبان اسمبلی نسبت به زبان های سطح بالا دشوارتر است. برنامه نویس باید به جزئیات توجه بیشتری نشان دهد و اطلاعات کافی نسبت به پردازنده مورد استفاده داشته باشد. اما برنامه های اسمبلی که ماهرانه نوشته شده باشند می توانند سریع تر و با حافظه کمتری از برنامه های مشابه نوشته شده با زبان سطح بالا اجرا شوند .

زبان ماشین

هر خانواده ای از پردازنده ها دارای مجموعه ای از دستورالعمل های منحصر بفرد است که زبان ماشین نامیده می شود. مجموعه دستورالعمل های یک پردازنده (Instruction Set) مجموعه ای از اعداد دودوئی است که ماشین می تواند آنها را درک و اجرا کند. هر نوع CPU تنها زبان مخصوص خود را درک می کند و دارای مفسری بنام microprogram است که دستورات زبان ماشین را به سیگنال های سخت افزاری تفسیر و ترجمه می کند .

مثال ۱. اعداد دودئی زیر یک دستورالعمل ماشین اینتل است که عدد 5 را در ثبات AL قرار می دهد .

1011 0000 0000 0101

مثال ۲. دستور زیر ثبات های EAX و EBX را جمع کرده و حاصل را در ثبات EAX ذخیره کند .

0000 0011 1100 0011

هر دستورالعمل زبان ماشین شامل کد منحصر بفردی دارد که کدعملیاتی (Operation Code) یا Opcode نامیده می شود . Opcodeهمیشه در ابتدای دستورالعمل قرار می گیرد. اکثر دستورات شامل داده هم هستند که توسط دستورالعمل استفاده می شود و عملوند (Operand) نام دارند .

کاملا واضح است که برنامه نویسی به زبان ماشین بسیار دشوار است. درک معنی دستورالعمل های کدشده زبان ماشین برای انسان کار خسته کننده ای است. خوشبختانه برای هر خانواده از پردازنده ها یک زبان اسمبلی ارائه می شود که دستورالعمل های زبان ماشین را به صورت نمادی و قابل فهم تر نشان می دهند .

زبان اسمبلی چیست؟

زبان اسمبلی که یک زبان برنامه نویسی سطح پایین است که ساختار و عملکردی وابسته به ماشین دارد. بین عبارات آن و دستورالعمل های زبان ماشین کامپیوتر تناظر یک به یک برقرار است. یعنی هر دستورالعمل اسمبلی دقیقا یک دستورالعمل زبان ماشین را نشان می دهد، در حالیکه در زبان سطح بالا یک عبارت معمولا به چندین دستورالعمل ماشین تبدیل می شود .

یک برنامه اسمبلی مانند برنامه های سطح بالا به صورت text نوشته می شود. هر دستورالعمل زبان اسمبلی یک نمایش نمادی (یک کد الفبائی کوتاه) از یک دستورالعمل ماشین است، که به این صورت معنی دستور واضح تر از کد زبان ماشین می شود .

مثال ۱. کلمه mov نمادی برای عمل انتقال داده است. دستور اسمبلی زیر جمع ثبات AL و عدد 5 را نشان می دهد.

```
mov AL,5
```

مثال ۲. کلمه add یک نماد برای دستورالعمل جمع است. دستور جمع ثبات های EAX و EBX به صورت زیر نوشته می شود.

```
add EAX, EBX
```

مشاهده می شود که به اینصورت درک معنی دستور بسیار روشن تر از کد ماشین معادل است .

مثال ۳. دستوری که عملوندی ندارد و فلگ carry را صفر می کند.

```
clc
```

مثال ٤. دستور زیر عدد یک را به ثبات AX اضافه می کند.

```
inc AX
```

مثال ٥. دستور جمع مقدار متغیر Count با محتوای ثبات به صورت زیر است.

```
mov AX,Count
```

هر دستور اسمبلی می تواند همراه با لیستی از عملوند ها باشد. فرم کلی دستورالعمل های اسمبلی به صورت زیر است :

```
mnemonic operand(s)
```

عملوند دستورالعمل می تواند از انواع زیر باشد :

- ثبات. عملوندهائی که مستقیما به محتوای ثبات های پردازنده مراجعه می کنند. مانند ثبات AL در مثال ۱ .
- متغیر یا حافظه ای .عملوندهائی که به داده ای در حافظه اشاره دارند. مانند متغیر Count در مثال ٥ .
- این عملوندها مقادیر ثابتی هستند که در داخل دستورالعمل قرار می گیرند. در مثال ۱عدد 5 یک عملوند فوری است .
ضمنی. عملوندهائی که صریحا در دستور ذکر نمی شوند. در مثال ٤عدد 1 با ثبات AL جمع می شود. عدد یک عملود ضمنی است.

اسمبلر

یک کامپیوتر نمی تواند مستقیما زبان اسمبلی را تفسیر کند و تنها قادر به اجرای کدهای زبان ماشین است. اسمبلر برنامه ای است که فایل متنی حاوی دستورات اسمبلی را خوانده و نمادهای اسمبلی را به کدهای زبان ماشین تبدیل می کند. البته کامپایلرها هم برنامه هائی هستند که عمل مشابه را برای زبان های سطح بالا انجام می دهند، اما اسمبلر به مراتب از کامپایلر ساده تر است، زیرا هرعبارت زبان اسمبلی تنها یک دستورالعمل ماشین را نشان می دهد. عبارات زبان سطح بالا پیچیده تر هستند و ممکن است به دستورالعمل های ماشین بیشتری نیاز داشته باشند.

یک تفاوت مهم دیگر بین اسمبلی و زبان های سطح بالا این است که هر نوع CPU زبان ماشین و زبان اسمبلی مخصوص به خود را دارد. انتقال برنامه های اسمبلی روی معماری های مختلف کامپیوتر به راحتی برنامه های سطح بالا نیست .

محبوب ترین اسمبلرها برای پردازنده ها ی خانواده اینتل عبارتند از:

- ماکرواسمبلر Microsoft's Assembler MASM
- توربو اسمبلر Borland's Assembler TASM
- و ASM86

برنامه دیگری که برای ردیابی اجرای برنامه و بررسی محتوای حافظه کاربرد دارد دیباگر (Debugger)است که استفاده از آن بهترین راه برای یادگیری برنامه های اسمبلی و روند اجرای آنهاست. دیباگر برنامه ای است که اجازه بررسی ثبات ها و حافظه را بعد از اجرای هر دستور برنامه می دهد و خصوصا برای تست برنامه های اسمبلی مفید است .

برنامه Debug از جمله ساده ترین دیباگرهاست که توسط MS-DOS عرضه شده است CodeView .همراه با میکروسافت اسمبلر می آید که اجازه می دهد کد منبع برنامه ها، بلاک های حافظه و ثبات ها را مشاهده کنید Turbo Debugger .بورلند هم به همین صورت است.

یک برنامه دیگر همراه با اسمبلر برنامه لینکر (Linker)است که فایل های مجزای تولید شده توسط اسمبلر یا کامپایلر را به یک برنامه اجرائی تبدیل می کند. برنامه Link که همراه فایل های MS-DOS می باشد یکی از متداولترین برنامه های لینکر می باشد .

هدف از یادگیري زبان اسمبلي

یادگیری زبان اسمبلی باید با فراگیری مفاهیم سیستم عامل و معماری کامپیوتر توام باشد تا به درک بهتر برنامه های اسمبلی و تعامل آن با کامپیوتر کمک کند. به چند دلیل ممکن است کسی بخواهد زبان اسمبلی را یاد بگیرد و از آن استفاده کند :

- زبان اسمبلی وسیله خوبی برای یادگیری نحوه کار کامپیوتر، کامپایلرها و زبان های سطح بالا است و به درک عمیق تر معماری کامپیوتر، مفاهیم سیستم عامل، نمایش داده ها و دستگاه های سخت افزاری کمک می کند که دانستن آنها باعث می شود برنامه نویس از عهده اشکالزدائی و رفع مسائل برنامه نویسی در سطح بالا بهتر برآید و نرم افزارهای پربارتری را در زبان های سطح بالا مانند C++ پیاده سازی کند.
- برنامه های اسمبلی سریع تر، کوچکتر و با توانائی های بیشتر از زبان های دیگر هستند و معمولا حافظه و زمان اجرای کمتری را نیاز دارد. گاهی نوشتن کد در اسمبلی سریعتر و کوتاهتر از کدکامپایل شده می شود. یک برنامه ویژوال بیسیک می تواند زیربرنامه های DLL نوشته شده در زبان اسمبلی را برای افزایش سرعت برنامه در حالات بحرانی فراخوانی کند.
- برنامه های اسمبلی می توانند براحتی از محدودیت های موجود در زبان های سطح بالا عبور کنند و کنترل بیشتری نسبت به نیازمندی های سخت افزاری خاص ارائه دهند. برخی از اعمال در زبان های سطح بالا دشوار یا غیر ممکن است، مانند ارتباط با سیستم عامل یا دسترسی مستقیم به کنترلرها. یک برنامه نویس مجرب می تواند با نوشتن کد بیشتر راهی برای گذشتن از این محدودیت ها پیدا کند اما خوانائی برنامه کاهش پیدا می کند. زبان اسمبلی، در مقابل، محدودیت های کمی دارد و تقریبا همه چیز را به نظر برنامه نویس واگذار می کند .

این دلایل نشان می دهند که فراگرفتن اسمبلی می تواند مفید باشد حتی اگر هیچوقت با آن برنامه ای نوشته نشود .

www.HPKClasses.ir

امروزه تولید برنامه ای که کاملا با زبان اسمبلی باشد غیر معمول است، زیرا برنامه نویسی در زبان سطح بالا بسیار ساده تر از اسمبلی است علاوه براین استفاده از اسمبلی باعث می شود حمل برنامه به کامپیوترهای مختلف دشوارتر شود. در حقیقت بندرت کسی کاملا در زبان اسمبلی برنامه می نویسد .در عوض اسمبلی برای بهینه سازی بخش های حساس برنامه و افزایش سرعت و دسترسی به سخت افزار و نوشتن برنامه های PROM استفاده می شود .

البته زبان برنامه نویسی C کیفیت منحصر به فردی در عرضه کردن مصالحه بین ساختار سطح بالا و جزئیات سطح پایین دارد. اکثر کامپایلرهای C توانائی تولید کد منبع اسمبلی را دارند. برنامه نویسان اغلب ترکیب C و اسمبلی را در برنامه های کاربردی به کار می برند .

www.HPKClasses.ir