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 های برنامه نویسی در دو سطح کلی طبقه بندی می شوند: زبان های سطح بالا و سطح پایین.

هر پردازنده ها قادر به درک و اجرای مجموعه ای از دستورالعمل های خاص است که زبان ماشین نامیده می شود. زبان ماشین در سطح پایین تنها زبان قابل درک پردازنده است که به صورت مجموعه ای از اعداد باینری است. مفسری بنام ریزبرنامه (Microprogram) دستورات زبان ماشین را به سیگنال های سخت افزاری تفسیر و ترجمه می کند.

اکثر برنامه نویسان با زبان های سطح بالا مانند C++ یا Java کار می کنند که هر عبارت آن به چند دستورالعمل زبان ماشین ترجمه می شود. نوشتن برنامه در زبان های سطح بالا خصوصا در محیط های ویژوال راحت تر، سریع تر و با هزینه کمتری انجام می شود.

درک معنی کدهای باینری زبان ماشین برای انسان کاری دشوار و خسته کننده است. به همین دلیل برای هر خانواده از پردازنده ها یک زبان اسمبلی ارائه شد که دستورالعمل های زبان ماشین را به صورت نمادی و قابل فهم تر نشان می دهند.

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

```
1011 0000 0000 0101
```

مثال ۲. کلمه mov نمادی برای عمل انتقال داده است. دستور اسمبلی زیر مقدار 5 را در ثبات AL قرار می دهد که معادل کد زبان ماشین مثال ۱ است.

```
mov AL,5
```

یک برنامه اسمبلی مانند برنامه های سطح بالا به صورت متنی نوشته می شود. هر دستورالعمل زبان اسمبلی یک کد الفبائی کوتاه (mnemonic) از یک دستورالعمل ماشین است، که به این صورت معنی دستور واضح تر از کد زبان ماشین می شود. بین عبارات زبان اسمبلی و دستورالعمل های زبان ماشین تناظر یک به یک برقرار است. یعنی هر دستورالعمل اسمبلی دقیقا یک دستورالعمل زبان ماشین را نشان می دهد و بالعکس، در حالیکه در زبان سطح بالا یک عبارت معمولا به چندین دستورالعمل ماشین تبدیل می شود.

مثال ۳. کلمه add یک نماد برای دستورالعمل جمع است. دستور جمع ثبات های EAX و EBX به صورت زیر نوشته می شود.

```
add EAX, EBX
```

کد زبان ماشین معادل دستور فوق به صورت زیر است:

```
0000 0011 1100 0011
```

مشاهده می شود که به اینصورت درک معنی دستورات اسمبلی بسیار روشن تر از کد زبان ماشین معادل است. با وجود این به دلیل وابستگی به ساختار پردازنده زبان اسمبلی یک زبان سطح پایین محسوب می شود. برنامه نویسی به 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. البته 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. به همین علت 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 از زبان اسمبلی استفاده می شود.

فرم کلی دستورالعمل های اسمبلی به صورت زیر است :

```
mnemonic operand(s)
```

هر دستور اسمبلی می تواند تعدادی عملوند به دنبال خود باشد. عملوند (operand) محل داده ای که قرار است دستورالعمل روی آن اجرا شود را مشخص می کند و می تواند به شکل های زیر باشد :

- **ثبات**. عملوندهائی که مستقیما به محتوای ثبات های پردازنده مراجعه می کنند. مانند ثبات AL در مثال ۲ .
- **متغیر** یا حافظه ای .عملوندهائی که به داده ای در حافظه اشاره دارند. مانند متغیر Count در مثال ٦ .
- **فوری**. این عملوندها مقادیر ثابتی هستند که در داخل دستورالعمل قرار می گیرند. مانند عدد 5 در مثال ۲.
- **ضمنی**. عملوندهائی که صریحا در دستور ذکر نمی شوند. در مثال ٥عدد ۱ با ثبات AL جمع می شود. عدد ۱ عملوند ضمنی است.

مثال ٤. دستوری که عملوندی ندارد و فلگ carry را صفر می کند.

```
clc
```

مثال ٥. دستور زیر عدد یک را به ثبات AX اضافه می کند.

```
inc AX
```

مثال ٦. دستور جمع مقدار متغیر Count با محتوای ثبات به صورت زیر است.

```
add AX,Count
```

اسمبلر

کامپیوتر تنها قادر به اجرای کدهای زبان ماشین است و نمی تواند مستقیما زبان اسمبلی را تفسیر کند بنابراین برنامه های زبان اسمبلی برای اجرا باید به زبان اسمبلی تبدیل شوند. این کار توسط اسمبلر صورت می گیرد. اسمبلر (Assembler) برنامه ای است که فایل متنی حاوی دستورات اسمبلی را خوانده و نمادهای اسمبلی را به کدهای زبان ماشین تبدیل می کند. اسمبلر مشابه کامپایلر عمل می کند اما به مراتب ساده تر است، زیرا هرعبارت زبان اسمبلی تنها یک دستورالعمل ماشین را نشان می دهد درحالیکه عبارات زبان سطح بالا پیچیده تر هستند و ممکن است به دستورالعمل های ماشین بیشتری نیاز داشته باشند. دقت داشته باشید هر پردازنده زبان ماشین و اسمبلی مخصوص خود را دارد و انتقال برنامه های اسمبلی روی معماری های مختلف کامپیوتر به راحتی برنامه های سطح بالا انجام نمی شود.

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

- ماکرواسمبلرMicrosoft's Assembler MASM
- توربو اسمبلرBorland's Assembler TASM

یک برنامه موردنیاز دیگر برنامه لینکر (Linker) است که فایل های مجزای تولید شده توسط اسمبلر یا کامپایلر را به یک برنامه اجرائی تبدیل می کند. برنامه link.exe که از فایل های سیستم عامل میکروسافت می باشد یکی از متداولترین برنامه های لینکراست.

هدف از یادگیری زبان اسمبلی

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 قابلیت حمل برنامه به کامپیوترهای مختلف را سخت تر می کند. در حقیقت بندرت کسی کاملا در زبان اسمبلی برنامه می نویسد. 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 پیاده سازی کند.
- برنامه های اسمبلی سریع تر، کوچکتر و با توانائی های بیشتر از زبان های دیگر هستند. گاهی نوشتن کد در اسمبلی سریعتر و کوتاهتر از کدکامپایل شده می شود. یک برنامه ویژوال می تواند زیربرنامه های DLL نوشته شده در زبان اسمبلی را برای افزایش سرعت برنامه در حالات بحرانی فراخوانی کند.
- برخی از اعمال در زبان های سطح بالا دشوار یا غیر ممکن است، مانند ارتباط با سیستم عامل یا دسترسی مستقیم به کنترلرها. برنامه های اسمبلی می توانند براحتی از این محدودیت ها عبور کنند.

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