

## آشنایی با زبان های برنامه نویسی سطح پایین و سطح بالا

پیش از شروع برنامه نویسی، می بایست با یک دسته بندی کلی در میان زبان های برنامه نویسی آشنا شد. در واقع، از یک بعد می توان **زبان های برنامه نویسی را به دو دسته ی سطح پایین و سطح بالا** تقسیم بندی کرد. زبان های برنامه نویسی سطح پایین به صورت مستقیم با پردازنده ی سیستم سر و کار دارند و با استفاده از آن ها می توان دستورات پایه ای برنامه نویسی را اجرا کرد. فهمیدن دستورات نوشته شده در این زبان ها کار دشواری است. به طور مثال، یک از این زبان های سطح پایین، **Machine Language یا زبان ماشین** نام دارد که به جای آن که از دستورات انگلیسی در آن استفاده شده باشد، از اعداد صفر و یک برای نوشتن دستورات در آن استفاده شده است. **زبان اسمبلی** هم یک نمونه ی دیگر از زبان های سطح پایین است.

برای آنکه بفهمیم زبان ماشین چگونه کار می کند، ابتدا می بایست درک کنیم که پردازنده ها چگونه کار می کنند. اساساً یک پردازنده از میلیون ها سویچ بسیار کوچک که می توانند خاموش و روشن شوند تشکیل شده اند. حال با خاموش یا روشن کردن برخی از این سویچ های خاص، می توان از پردازنده خواست تا کار خاصی را انجام دهد.

به جای روشن یا خاموش کردن این سویچ ها به صورت دستی، زبان ماشین یا همان صفر و یک این امکان را به شما می دهد تا با ۰ و ۱ که ۰ به معنی خاموش و ۱ به معنی روشن است، این سویچ ها را خاموش و روشن کنید.

با توجه به این که نوشتن برنامه های کامپیوتر به زبان باینری یا همان زبان ماشین، کاری بسیار طاقت فرسا بوده و از سوی دیگر احتمال بروز اشتباه هم در آن زیاد است، دانشمندان علوم کامپیوتری زبانی تحت عنوان اسمبلی را طراحی کردند.

در یک کلام، **هدف اصلی زبان اسمبلی این بوده است** تا برنامه نویسی و یا بهتر بگوییم صحبت کردن با سی پی یو را راحت تر سازد. برای آن که درک کنیم که ساز و کار زبان اسمبلی به چه شکل است، ابتدا می بایست بفهمیم که پردازنده ها چگونه داده ها را پردازش می کنند. توجه داشته باشیم که پردازنده ی سیستم های کامپیوتری همانند «مغز» آن کامپیوتر است. همان طور که مغز انسان ها زمانی که در ارتباط با سایر اعضای بدن و جهان واقع باشد می تواند کارایی داشته باشد، پردازنده ی کامپیوتری هم صرفاً زمانی کار خواهد کرد که با سایر بخش های سیستم در ارتباط بوده و بتواند با دنیای بیرون -یا همان برنامه نویس- در ارتباط باشد. راه های ارتباطی که یک پردازنده از آن طریق با بخش های دیگر سیستم در ارتباط است اصطلاحاً **باس** گفته می شود.



فرض کنیم که یک پردازنده می‌خواهد با دیتای خاصی کار کند. در این صورت، پردازنده داده‌ها را از بخش دیگر سیستم مثلاً هارد دیسک گرفته و به صورت موقت آن‌ها را در مکانی که اصطلاحاً **رجیستر** گفته می‌شود ذخیره می‌سازد. سپس پردازنده تغییرات را روی داده‌ها اعمال کرده و داده‌های تغییر یافته را به بخش دیگری از سیستم مثل حافظه ارسال می‌کند. به عبارت دیگر، همان‌طور که در اولین کامپیوترهای ساخته شده برای اجرای یک دستور جای کابل‌ها و سوییچ‌ها عوض می‌شد، در کامپیوترهای امروزی جای صفرها و یک‌ها تغییر می‌یابد که این تغییر جایگاه صفر و یک‌ها با استفاده از زبان ماشین انجام می‌پذیرد.

اگرچه که زبان اسمبلی به مراتب از زبان ماشین -یا همان صفر و یک- راحت‌تر است، اما به هر حال برای ساخت نرم افزارهای بزرگ و پیچیده زبانی دشوار برای برنامه نویسان محسوب می‌شود. در گذشته، بسیاری از نرم‌افزارها با استفاده از این زبان نوشته می‌شد اما به مرور که برنامه‌ها پیچیده‌تر شدند، ثابت شد که زبان اسمبلی زبان اثربخشی برای کدنویسی نیست!

بزرگ‌ترین مشکل زبان اسمبلی این است که برای نوشتن یکسری دستورات با استفاده از این زبان، می‌بایست با رجیسترهای پردازنده که پیش از این با آن‌ها آشنا شدیم کار کنیم. به عبارت دیگر، به منظور جمع کردن دو عدد با یکدیگر، برنامه نویسی می‌بایست به پردازنده دستور دهد تا یک عدد را در یک رجیستر ذخیره سازد، سپس عدد دوم را به عددی که در رجیستر ذخیره شده اضافه کند و در نهایت نتیجه را از رجیستر بازخوانی کند.

نوشتن یک برنامه به زبان ماشین -حتی اگر آن برنامه خیلی هم ساده باشد- کار نسبتاً دشواری است. بسیاری از برنامه نویسان از زبان اسمبلی در جاهایی که نیاز به سرعت و اثربخشی بالا دارند استفاده می‌کنند اما توجه داشته باشیم که برنامه‌های نوشته شده با استفاده از زبان اسمبلی به مراتب کندتر از برنامه‌های نوشته شده با زبان ماشین هستند چرا که برای اجرا، برنامه‌های نوشته شده با زبان اسمبلی ابتدا می‌بایست به زبان ماشین تفسیر شوند که معمولاً این کار توسط برنامه‌هایی تحت عنوان **Assembler** انجام می‌شود که این اسمبلر کارش این است که یک برنامه‌ای که با استفاده از زبان اسمبلی نوشته شده باشد را گرفته و آن را تبدیل به زبان ماشین یا همان صفر و یک کند.

توجه داشته باشیم که هر پردازنده ای صرفاً زبان اسمبلی خاص خود را می‌فهمد. بنابراین یک پردازنده ی **Intel Core ۲** زبان اسمبلی پردازنده **PowerPC** را نمی‌فهمد و بالعکس. به هر حال برخی پردازنده‌ها هم هستند که با سایر پردازنده‌ها تعامل خوبی دارند مثل پردازنده‌های **AMD** که مثلاً با پردازنده‌های اینتل سازگار می‌باشند.

این سر و کله زدن با رجیسترهای پردازنده، منجر به سردرگمی بیش از پیش برنامه نویسان و کسانی که علاقمند به برنامه نویسی بودند شد لذا دانشمندان علوم کامپیوتری به فکر طراحی زبان‌های برنامه نویسی سطح بالا افتادند که در ادامه بیشتر با آن‌ها آشنا خواهیم شد.



در مقابل زبان های برنامه نویسی سطح پایین یا Low Level، زبان های برنامه نویسی سطح بالا از دستوراتی همچون کلمات انگلیسی - که برای انسان ها قابل فهم تر هستند - استفاده می کنند. زمانی که برنامه نویسی دستورات مد نظر اش را در یکی از این زبان های سطح بالا - همچون زبان های پایتون، جاوا اسکریپت، پی اچ پی، روبی، و غیره - می نویسد، یک نرم افزار واسطه ای می آید که آن کدها را به زبان ماشین - یا همان صفر و یک - ترجمه کرده و در اختیار سیستم قرار می دهد چرا که سیستم ها فقط و فقط معنی زبان ماشین یا همان صفر و یک را متوجه می شوند.

هدف اصلی طراحی زبان های High-level یا همان سطح بالا این بوده است تا فرایند برنامه نویسی راحت تر گردد. بنابراین به جای این که به سیستم دستور دهیم تا عدد ۲ را در رجیستر ذخیره سازد، سپس عدد ۳ را به آن اضافه کند و در نهایت خروجی آن را بگیرد، زبان های برنامه نویسی سطح بالا این امکان را در اختیار برنامه نویس قرار می دهند تا به سیستم هر دستوری که می خواهند بدهند و اصلا کاری به این که سیستم قرار است به چه شکل آن دستور یا دستورات را عملی سازد نداشته باشند.

زبان های برنامه نویسی سطح بالا همچون فورتران، بیسیک، کوبول و پاسکال منجر به این شدند تا برنامه نویسان وارد جزئیات برنامه نویسی و نحوه ی کار پردازنده نشوند اما این دور بودن از جزئیات کار منجر به این می شد تا در برخی جاها دست برنامه نویسان آن طور که باید و شاید باز نباشد بنابراین برای آن که فصل مشترکی مابین زبان های سطح بالا و زبان اسمبلی - که یک زبان سطح پایین است - ایجاد شود، زبانی تحت عنوان زبان برنامه نویسی C ابداع شد.

ایده ی پشت این زبان فراهم آوردن فرصتی برای برنامه نویسان بود تا از آن طریق امکان صحبت کردن با سی پی یو - یا همان مغز کامپیوتر - به صورت مستقیم همچون زبان اسمبلی را داشته باشند اما در عین حال این فضا را هم برای برنامه نویس فراهم آوردند تا بتوانند در صورت نیاز، جزئیات فنی نحوه ی کار سی پی یو را هم همچون یک زبان سطح بالا نادیده بگیرند.

زبان برنامه نویسی C این امکان را به برنامه نویسان می دهد تا پردازنده را همچون زبان اسمبلی کنترل کنند اما در عین حال برنامه هایی بنویسند که برای انسان ها قابل فهم و قابل خواندن و نوشتن باشند. بسیاری از برنامه های کاربردی که امروزه می بینیم مثل سیستم عامل های ویندوز، لینوکس، مکینتاش و ... با استفاده از این زبان نوشته شده اند و زبان C یک زبان مادر در دنیای برنامه نویسی محسوب می شود.

با این تفاسیر، دیگر نیازی به توضیح نیست که چرا بسیاری از برنامه های کاربردی دنیا با استفاده از زبان برنامه نویسی سی نوشته می شوند (در واقع این زبان هم خوبی های زبان اسمبلی در سرعت را داشته و هم خوانایی زبان های سطح بالا را برای ویرایش و توسعه ی نرم افزار توسط سایر برنامه نویسان را دارا است).

نکته



در پاسخ به این سؤال که سریع ترین زبان برای صحبت کردن با سی پی یو آیا زبان ماشین، زبان اسمبلی، زبان های سطح بالا یا زبان سی است، بایستی گفت که مسلماً زبان ماشین -یا همان صفر و یک- سریع ترین زبان برنامه نویسی دنیا است چرا که سیستمها فقط و فقط زبان ماشین را می فهمند و به نوعی می شود گفت که زبان ماشین، زبان مادری آنها است!

برای این که پردازنده ها برنامه هایی که با استفاده از زبان های سطح بالا نوشته می شوند را هم بفهمند، دانشمندان برنامه ی واسطی تحت عنوان **Compiler (کامپایلر)** طراحی کرده اند تا کدهای زبان های سطح بالا را به زبان ماشین یا زبان باینری یا همان زبان صفر و یک مبدل سازند تا برای کامپیوتر قابل فهم شوند.

### عموماً زبان های برنامه نویسی را به پنج نسل تقسیم می کنند:

- نسل اول **زبان ماشین** است همان زبان صفر و یک
- نسل دوم زبان هایی مانند اسمبلی و مشتق آن است که قابل فهم تر برای انسان می باشد
- نسل سوم زبان هایی مانند کوبول و پی ال وان و... است که دستورهای قابل فهم تر برای انسان و نیاز به کامپایلرها
- نسل چهارم مثل زبان های اوراکل و **فاکس پرو** و اس کیو ال ها است و این نسل چیزی نزدیک به محاوره های انسانی است.
- نسل پنجم زبان هایی مانند **prolog , ops**، **ویژوال بیسیک** - تمرکز بر **حل مسئله** و استفاده از الگوریتم های نوشته شده توسط برنامه نویس

### زبان های برنامه نویسی را می توان از چهار دیدگاه متفاوت مورد بررسی قرار داده و تقسیم بندی کرد:

الف) روش های برنامه نویسی

۱. زیر روالی
۲. ساخت یافته
۳. مدولار
۴. شیء گرا

ب) نزدیکی به زبان ماشین

۱. سطح پایین
۲. سطح میانی

۳. سطح بالا

(ج) نوع ترجمه

۱. مفسری

۲. کامپایلری

(د) رابط برنامه نویسی

۱. مبتنی بر متن

۲. مبتنی بر گرافیک (ویژوال)

**دستور زبان** برنامه نویسی معمولاً به وسیله ترکیب عبارات معین (برای ساختار لغوی) و فرم توضیح اعمال (برای ساختار گرامری) تعریف می‌شوند. متن زیر یک گرامر ساده، به زبان lisp است:

expression ::= atom | list

atom ::= number | symbol

+['۹'-'۰']?[-+] ::= number

\*./symbol ::= ['A'-'Za'-'z

'list ::= '(' expression\* ')

## سؤالات درس جلسه دوم

- ۱- انواع زبانهای برنامه نویسی را نام ببرید؟
- ۲- زبانهای برنامه نویسی سطح بالا و سطح پایین را با هم مقایسه کنید؟
- ۳- چند نمونه از زبانهای برنامه نویسی سطح بالا و سطح پایین را نام ببرید؟
- ۴- زبان اسمبلی یا زبان ماشین چگونه کار می کند؟
- ۵- کامپایلر را در برنامه نویسی توضیح دهید؟
- ۶- انواع نسل های برنامه نویسی توضیح دهید؟
- ۷- تقسیم بندی زبان های برنامه نویسی از دیدگاه متفاوت را نام ببرید؟