فصل دوم

پردازنده های زبانهای برنامه سازی

by: Mahdi Sadeghizade
Email: Mahdi_sadeghe@yahoo.com
Site: www.Sadeghizadeh.ir

به طور مختصر می توان کامپیوتر ها را مجموعه ای از الگوریتم ها و داده ها که قادر به ذخیره و اجرای برنامه است ، دانست . یک کامپیوتر به دو روش کاملا سخت افزاری و دیگری ترکیبی از سخت افزار و نرم افزار می تواند ساخته شود .

در توسعه زبانهای برنامه سازی سه عامل بر روی طراحی زبان مؤثر هستند :

1. کامپیوتر سخت افزاری :
   کامپیوتری که برنامه های نوشته شده به آن زبان باید به روی آن اجرا شوند .
2. کامپیوتر نرم افزاری :
   مدل اجرا و یا کامپیوتر مجازی که آن زبان را بر روی سخت افزار واقعی پشتیبانی می کند .
3. مدل محاسباتی :
   که زبان آن را پیاده سازی می کند .

کامپیوتر های سخت افزاری (Hardware) و میان افزاری (Firmware)

منظور از کامپیوتر سخت افزاری کامپیوتری است که کاملا از اجزای سخت افزاری و مدارات الکتریکی ساخته شده است . وضعیت ابتدایی برنامه ، مطابق Initial State و نتیجه نهایی برنامه Final State کامپیوتر است.
کامپیوتر به صورت میان افزاری ساخته می شود در صورتی که هر دستور زبان ماشین برابر دنباله ای از ریز عمل ها (Micro Operation)که در حافظه قابل برنامه ریزی ذخیره شده است .

ساختار عملیات یک کامپیوتر :

یک کامپیوتر از دید ساختار زبان برنامه سازی :

- **Data :** یک کامپیوتر باید مجموعه ای از داده اولیه و ساخت یافته را برای انجام عملیات مختلف فراهم آورد (داده از یک جزء ، داده اولیه است اما اگر از چند جزء داده تشکیل شده باشد داده ساخت یافته است)
- **Primitive Operation** : یک کامپیوتر باید مجموعه ای از عملیات اولیه برای عملیات روی داده ها را در برداشته باشد مانند دستورات CPU یا زبان ماشین .

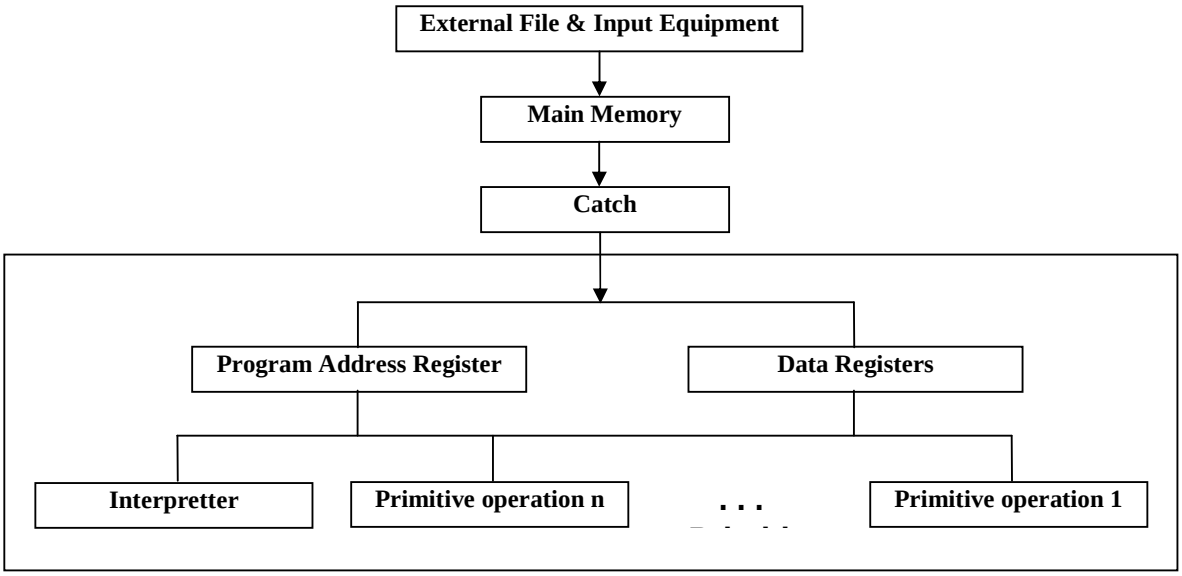
☑ Primitive Operation دستورات زبان ماشین هستند که یکی از آنها اجرا می شود

- **Data Control** : یک کامپیوتر باید مکانیزم هایی را جهت قرار دادن اطلاعات به عنوان ورودی عملیات فراهم سازد که نسبت دادن آدرس به هر خانه حافظه و همچنین برای ثبات ها در این بخش قرار دارد .

- **Sequence Control** : یک کامپیوتر باید مکانیزم هایی را جهت کنترل دستورات فراهم کند .
- **Storage Management** : یک کامپیوتر باید مکانیزم هایی را برای کنترل تخصیص حافظه برای برنامه و داده ها و همچنین آزاد سازی حافظه داشته باشد .
- **Operation environment** : یک کامپیوتر باید مکانیزم هایی را برای مبادله اطلاعات با دستگاههای جانبی و ورودی / خروجی (I/O) فراهم نماید .

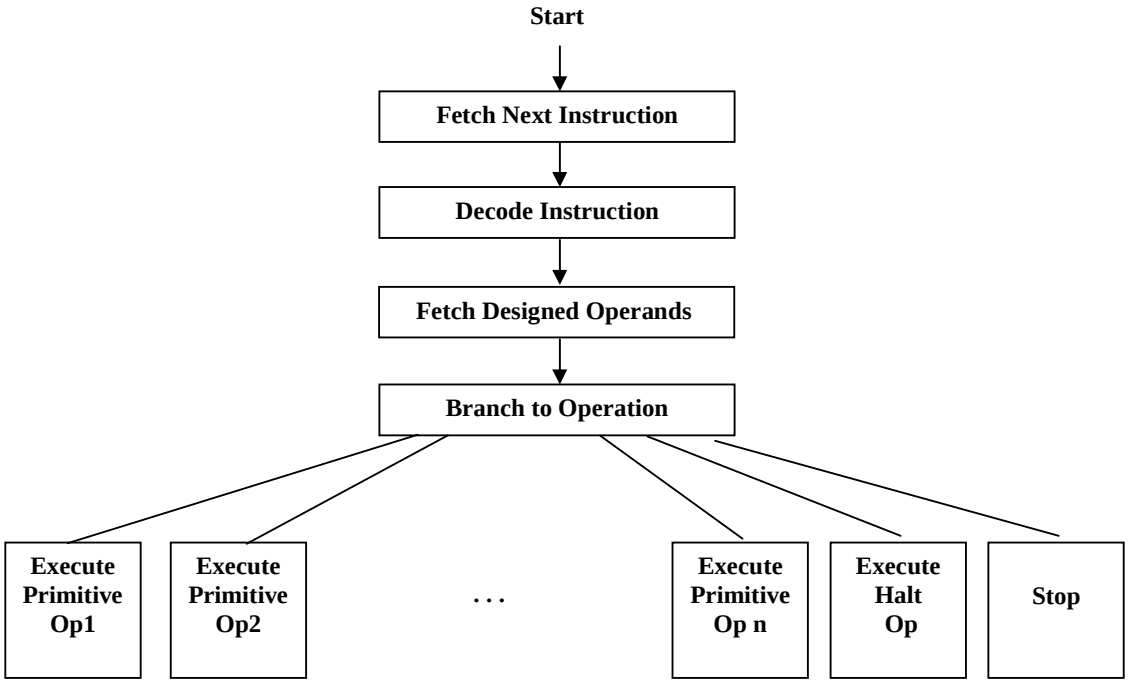
☑ کامپیوتر دارای چند نوع داده پیش ساخته (توکار) است که مستقیما توسط سخت افزار دستکاری شوند که عبارتند از : صحیح ، حقیقی دقت معمولی (یک کلمه) ، رشته هایی با طول ثابت و رشته های بیتی با طول ثابت .

در یک دید ، ساختمان کامپیوتر را می توان به شکل زیر در نظر گرفت :



سطح چهارم نمودار Active Processing Element هستند ، این عناصر برای انجام پردازش مورد نظر فعال می شوند .

سیکل اجرای دستورات :



انواع کامپیوترها بر اساس اعمال اولیه :

کامپیوترهایی با مجموعه دستورات کاهش یافته RISC

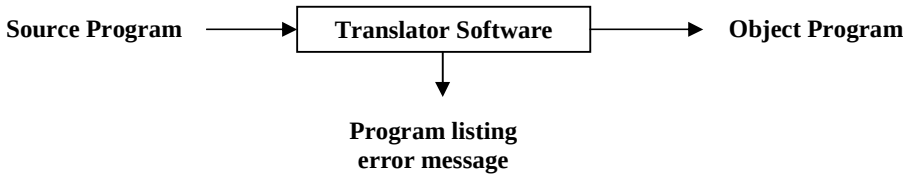
کامپیوترهایی با مجموعه دستورات پیچیده CISC

کامپیوتر های مترجم و شبیه سازی شده :

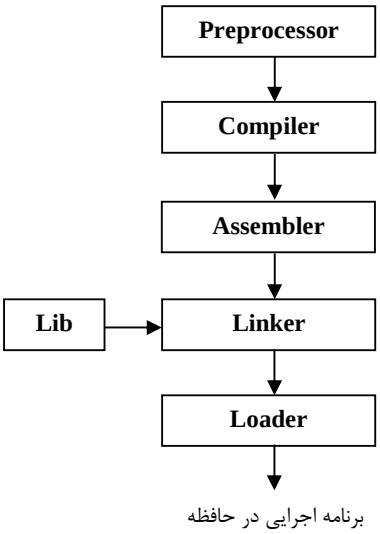
برای پیاده سازی و اجرای یک زبان سطح بالا می توان از طریق دو راه زیر عمل نمود :

1. Translation (Compilation)

در این روش برنامه به زبان سطح بالا طی فرایندهایی تبدیل به زبان ماشین می شود که قابل اجرا روی سخت افزار است .



انواع مترجم ها



- Preprocessor : منظور از Preprocessor آن است که ورودی ، برنامه سطح بالا و خروجی آن به صورت استاندارد هایی که همان زبان می باشد مانند فایل های Include در زبان C .
- Compiler : ابزاری است برای انتقال برنامه به شکلی که بتواند بر روی سخت افزار اجرا شود ، استفاده می شود.
- Assembler : انتقال برنامه از اسمبلی به زبان ماشین را انجام می دهد .
- Linker : قسمت های مختلف برنامه را به هم متصل کرده و برنامه را قابل اجرا می کند .
- Loader : برنامه قابل اجرا را در حافظه قرار می دهد .

☒ عیب اصلی این روش از دست دادن اطلاعاتی راجع به برنامه است اگر داده هایی از برنامه منبع به دنباله هایی از آدرس زبان ماشین ترجمه شوند و خطایی در برنامه وجود داشته باشد (مثلا تقسیم بر صفر) به سختی می توان تعیین کرد که کدام دستور در اجرا بوده است و کدام اشیاء داده دست کاری شده اند .

2. Software Interpretation:

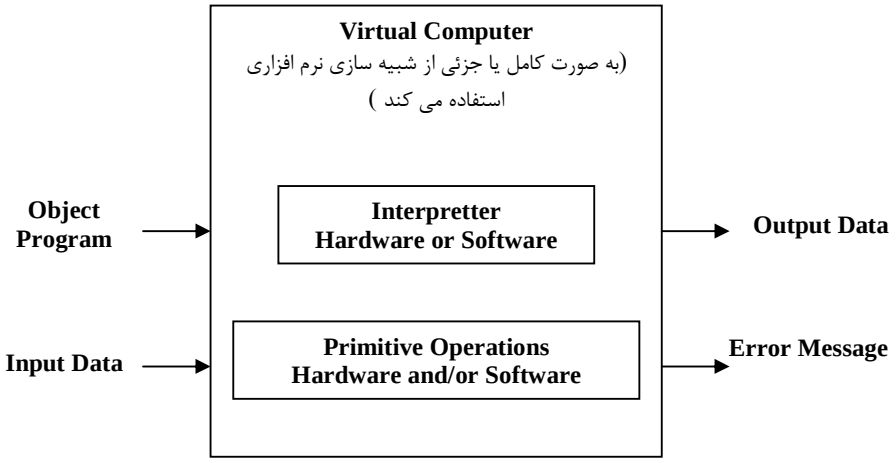
در این مکانیزم Source برنامه مستقیما به شبیه ساز داده می شود (که شبیه ساز یک نرم افزار است) و شبیه ساز دستور زبان سطح بالا را تفسیر و مجموعه دستورات لازم را برای انجام دستور زبان سطح بالا اجرا می کند .

تفاوت این دو روش در آن است که در روش اول برنامه کامل به زبان ماشین تبدیل شده و سپس اجرا می شود در حالی که در روش دوم تک تک دستورات زبان سطح بالا ابتدا تفسیر شده و مجموعه دستورات لازم برای شبیه سازی آن دستور اجرا می شود .

- ☑ روش اول و دوم به صورت محض کمتر به کار می رود بلکه در عمل از ترکیبی از آنها به صورت مؤثر استفاده می شود .
- ☑ یک کامپیوتر مجازی کامپیوتری است که با زبانی غیر از زبان ماشین کار می کند .
- ☑ به عنوان یک قاعده کلی ترجمه وقتی استفاده می شود که ساختار زبان منبع نمایش مستقیمی در زبان مقصد داشته باشد که در این حال بسط کد چندان مشکل نخواهد بود و در بقیه موارد از شبیه سازی استفاده می شود .

یک سئوال کلیدی آن است که آیا نمایش اصلی برنامه در ضمن اجرا همان نمایش زبان ماشین کامپیوتر واقعی است یا خیر ؟ که بر این اساس دو نوع تقسیم بندی برای زبان ها وجود دارد :

1. زبان های کامپایلری : C,C++,Pascal,Fortran,Ada
2. زبان های مفسری : Lisp , ML,Prolog,Smalltak,Perl,Postscript
   - زبان های بین 2 حالت بالا : WWW , Java



پیاده سازی زبان های کامپیوتر مجازی :

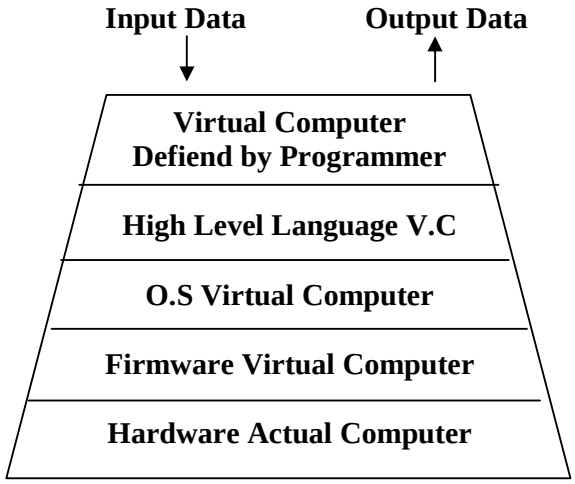
با توجه به تعریف قبلی در مورد کامپیوتر مجازی می توان راههای مختلفی را برای ساخت یک کامپیوتر به صورت زیر طبقه بندی نمود :

1. از طریق سخت افزار
2. از طریق نرم افزار
3. ترکیبی از سخت افزار و نرم افزار : میان افزار
4. از ترکیب همه سه نوع

با توجه به موارد فوق عواملی که پیاده سازی یک زبان سطح بالا را مختلف می سازند عبارتند از :
- اختلاف در امکاناتی که روی کامپیوتر پایه وجود دارد .
- اختلافاتی که در مفاهیم پیاده سازی VC وجود دارد که به طور ضمنی در تعریف زبان ملموس است .
- اختلاف در انتخاب هایی که برای پیاده سازی و شبیه سازی زبان های سطح بالا می تواند به کار گرفته شود .

سلسه مراتب کامپیوتر ها :

از دید برنامه نویس به زبان سطح بالا ، ساختار یک VC را می توان به صورت سلسه مراتب زیر تصور کرد :



لایه تحتانی کاملا از اجزای فیزیکی ساخته شده است و بقیه لایه ها توسط لایه پایین تر به اجرا در می آیند.
در صورتی که روش اول برای پیاده سازی انتخاب شود لایه فوقانی دوم یا همان کامپیوتر مجازی میان افزار وجود ندارد یا به صورت ضعیف وجود دارد در صورتی که از روش دوم همه لایه ها را خواهیم داشت و همه لایه ها عمل خواهند کرد .

Binding و زمان Binding (انقیاد ، مقید سازی)

به هنگام پیاده سازی و اجرای یک برنامه به زبان سطح بالا یک عنصر از برنامه می تواند یک خصوصیت از مجموعه خواص ممکن را به خود بگیرد که این عمل را Binding و زمان آنرا Binding Time می گویند .
مثلا یک متغیر صحیح در حین اجرا می تواند یک ارزش از ارزش های ممکن را به خود بگیرد .

انواع مختلف Binding

انوع Binding بر حسب زمان انجام این عمل طبقه بندی شده و شامل موارد زیر است :

1. در زمان اجرا (RunTime/Execution Time)

در این نوع Binding دو حالت وجود دارد :

-در ورود به زیر برنامه یا بلاک (Block): مثلا در C و Pascal انقیاد پارامتر های مجازی به واقعی و انقیاد پارامترهای مجازی به محل های حافظه
-در نقطه خاصی از اجرای برنامه : مثلا انقیاد متغیر ها به مقادیر شان توسط دستور انتساب

☑ در زبانهایی مثل Lisp و ML انقیاد اسامی به محلهایی از حافظه در هر نقطه ای از برنامه صورت می گیرد.

2. در زمان ترجمه (Compile Time/Translatio Time)

- انقیاد انتخاب شده و یا اعمال شده **توسط برنامه نویس** مانند تعریف نوع متغیر ها و همچنین تعریف ساختار برنامه .
- انقیاد انتخاب شده و یا اعمال شده **توسط مترجم** . مثلا محل نسبی شی داده در حافظه ای که برای زیر برنامه اختصاص داده شده است . شکل ذخیره آرایه ها و چگونگی ایجاد توصیف گرها ، آرایه
- انقیاد انتخاب شده و یا اعمال شده **در زمان Load برنامه** . مثلا انقیاد متغیرها به محل های واقعی حافظه یا به عبارت دیگر انقیاد های مربوط به گرفتن هر نوع حافظه ای برای اشیا داده ای داخل برنامه .

3. در زمان پیاده سازی زبان:

این انقیاد حالت عملی و پیاده سازی دارد مانند پیاده سازی عملگرها ، نمایش اعداد در حافظه ، انجام عملیات و محاسبات ریاضی و همچنین رینج مقداری متغیر ها در این قسمت قرار دارد .

4. در زمان تعریف زبان:

اغلب ساختارهای زبان برنامه سازی در هنگام تعریف زبان تعیین می گردد . به عنوان مثال شکل های مختلف دستورات انواع ساختمان داده ها ، ساختارهای برنامه و ...

☑ پاسخ سئوالات زیر در زمان تعریف زبان می باشد :
چه انواعی داریم ؟ ثابت ها کدامند ؟ عملگرها کدامند ؟ شکل ظاهری عملگرها
Syntax دستورات؟ ساختار برنامه؟ فرم دستورات و عملی که هر دستور انجام می دهد ؟
توابع از پیش تعریف شده (توابع کتابخانه ای)

به عنوان مثال ، متغیرهایی که با حرف i و j در زبان فرترن شروع می شوند همگی از نوع Integer هستند .

زمان انقیاد می تواند روی انعطاف پذیری و سرعت برنامه مؤثر باشد . اگر این عمل در حین اجرا انجام شود انعطاف پذیری زیاد می شود ولی سرعت نیز کم می شود و اگر این عمل در زمانی غیر از زمان اجرا باشد سرعت اجرای برنامه زیاد است ولی انعطاف پذیری کم خواهد بود .
یک دلیل برای توجیح این واقعیت این است که اگر نوع یک متغیر در حین اجرا بتواند تغییر نماید انعطاف پذیری زیادی به وجود می آید بنابراین سرعت اجرای برنامه به دلیل استفاده از روتین های تبدیل نوع اطلاعات کند می شود .

انواع زبان ها بر اساس زمان مقید سازی

1. زبانهایی با انقیاد زودرس (EBT) : کارایی بالا ، سرعت بالا ، انعطاف
   مانند زبان های Fortran , C , Pascal و ... که انقیاد در آنها در زمان ترجمه انجام می شود . (معمولا کامپایلری است)
2. زبان هایی با انقیاد دیر رس (LBT):
   مانند : Basic , Prolog , Lisp , ML که اغلب انقیاد در آنها در زمان اجرا انجام می شود . (معمولا مفسری هستند)

☑ در زبانی مثل Ada که هم برای قابلیت انعطاف و هم برای کارایی طراحی شده می توان زمان انقیاد را انتخاب کرد .

مثال : زمانهای انقیاد دستور مقابل در زبان پاسکال :

x:=x+10

1. مجموعه انواع ممکن برای متغیر x : مثلا Real باشد یا Integer که در زمان تعریف مشخص می شود .
2. نوع خود متغیر x : در زمان ترجمه مشخص می شود (مثلا با دستور int x) در برخی از زبانها مثل Smalltalk و Prolog نوع متغیر ممکن است در زمان اجرا مشخص شود .
3. مجموعه ای از مقادیر ممکن برای x : که در زمان پیاده سازی زبان تعیین می شود .
4. مقدار متغیر x : در هر نقطه ای از اجرای برنامه مقدار خاصی به متغیر x مقید می شود .
5. نمایش مقدار ثابت 10 : انتخاب نمایش دهدهی در برنامه یعنی نمایش خود " 10 " در زمان تعریف انجام می شود و انتخاب دنباله ی بیتی آن در زمان اجرا و نحوه ی پیاه سازی آن در زمان پیاده سازی مشخص می شود .
6. خصوصیات عملگر + : انتخاب نماد "+" برای نمایش عمل جمع در زمان تعریف زبان صورت می گیرد . در زمان کامپایل مشخص می شود که چه عملی را انجام می دهد . مثلا جمع صجیج (که ممکن در زمان اجرا نیز این عمل انجام شود) و پیاده سازی هر عمل جمع در زمان پیاده سازی انجام می شود .