تعریف کامپیوتر

سخت افزاری

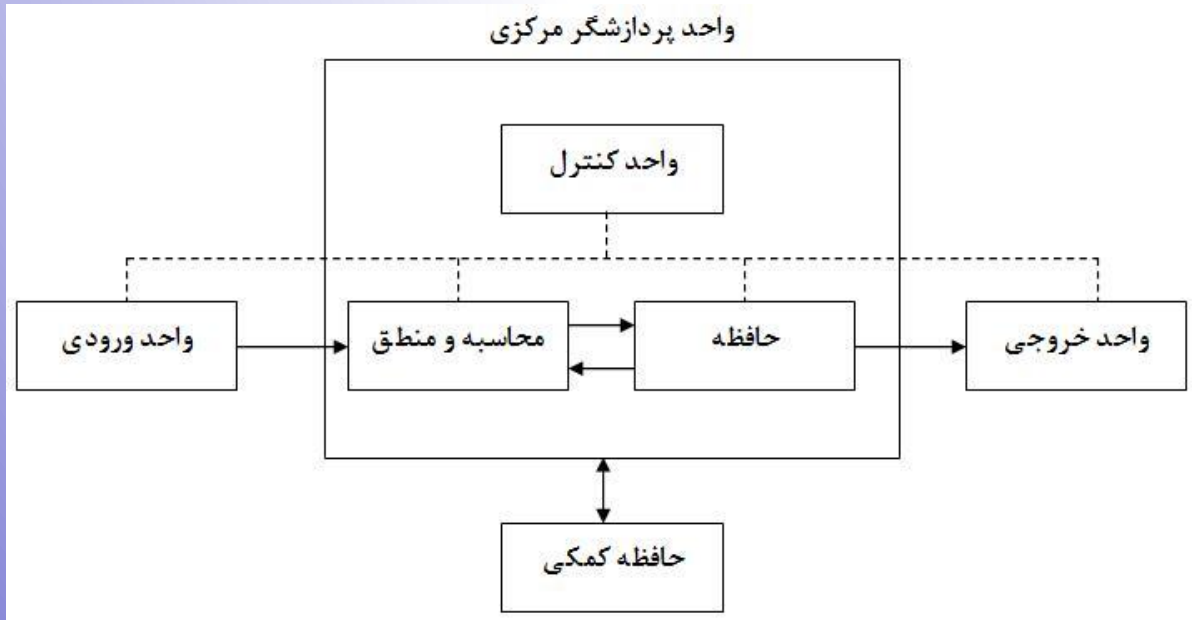
- متشکل از قطعات الکترونیکی

برای انجام محاسبات به صورت

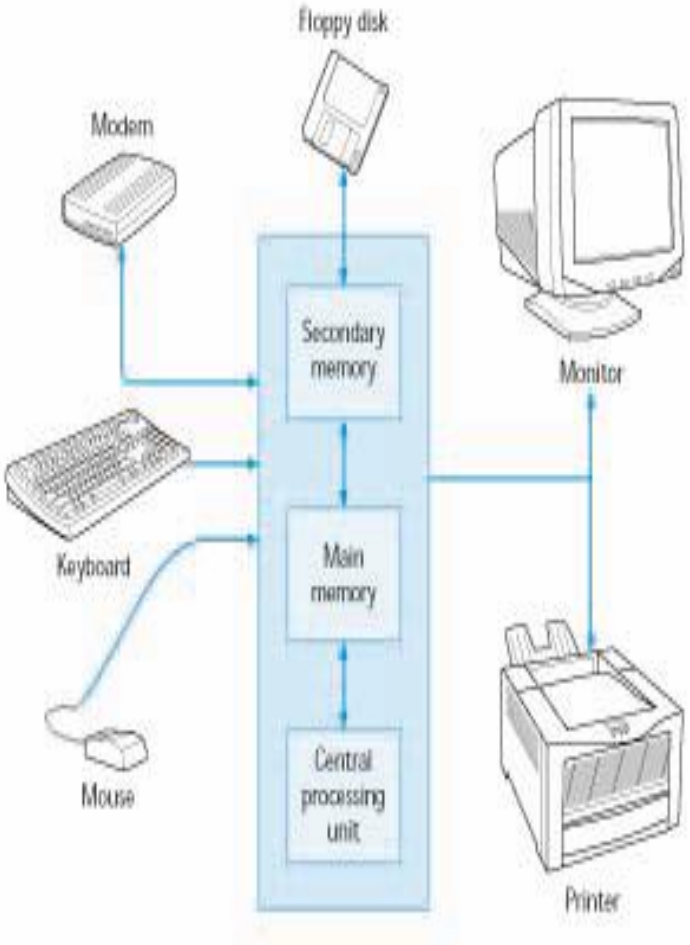
- پذیرفتن و ذخیره سازی داده ها و دستورات
- اجرای سریع و دقیق دستورات بر روی داده ها
- امکان تحویل نتایج و اطلاعات به انسان

بخش های مختلف کامپیوتر

- واحد ورودی (Input unit)
- واحد خروجی (Output unit)
- واحد حافظه (Memory unit)
- واحد محاسبه و منطق (ALU)
- واحد کنترل (Control unit)



اجزای سخت افزاری کامپیوترها



- CPU (واحد پردازش مرکزی) (مغز کامپیوتر)
  - ALU، کنترل درخواستهای ورودی/ خروجی
- Main Memory
  - ذخیرهٔ داده های مورد نیاز جهت پردازش
  - انواع آن: RAM و ROM
- Secondary Memory
  - ذخیره سازی داده ها و برنامه ها
  - انواع آن: دیسک، دیسک سخت، CD
- Input Devices
- Output Devices
- Network Connction

مقایسه انسان و کامپیوتر

- دقت و صحت کار و قابلیت اعتماد:
  - عملکرد یکسان در دفعات اجرای متفاوت و صحت و دقت عملکرد در صورت صحت داده ها و دستورالعملهای ورودی در کامپیوتر
  - انسان در دفعات و شرایط مختلف (جسمی و روحی) ممکن است عملکردهای متفاوتی داشته باشد.

- نحوهٔ ذخیره سازی اطلاعات و سرعت دستیابی به آنها
  - دریافت اطلاعات از طریق حواس در انسان و از طریق دستگاه های ورودی در کامپیوتر
  - قدرت ذخیره سازی بالای اطلاعات در کامپیوتر
  - پردازش اطلاعات
    - در کامپیوتر: انجام میلیون ها دستورالعمل در کسری از ثانیه
    - در انسان: قدرت شناسایی تصویر در کسری از ثانیه (در کامپیوتر چندین سال طول می کشد)

داده های کامپیوتر

- داده های مورد استفاده انسان
  - ارقام ۴، ۵، ۹
  - حروف A، B، Z
  - علائم !، ^، ٪
- کامپیوتر
  - استفاده از نمایش دودویی و باینری (تبدیل تمام داده ها به ۰ و ۱)
  - یک بیت: اطلاعات صفر یا یک
  - یک بایت: مجموعه ای از ۸ بیت

0	0	0	0	1	0	0	1

0	0	1	0	0	0	1	0

آشنایی با نرم افزار کامپیوتر

- نرم افزار های کاربردی
  - برنامه هایی که برای موضوعات و کاربردهای خاص کاربر نوشته شده اند.
  - باعث فراگیر شدن استفاده از سیستم می شوند.
  - برخی نرم افزارهای کاربردی عمومی:
    - Word Processors
    - Presentation Managers
    - Drawing Programs

آشنایی با نرم افزار کامپیوتر -۲

- نرم افزارهای سیستمی
  - برنامه هایی که اجرای سایر برنامه ها را پشتیبانی می کنند.
  - انواع نرم افزارهای سیستم:
    - سیستم های راه انداز (Operating Systems)
      - مرتب کردن سیستم
      - توانایی کنترل ورودی / خروجی با تعداد زیادی از دستگاه ها
      - مثال: MSDOS، Windows، Unix
    - سیستم های مفسر (Translation Systems)

تاریخچه زبان های برنامه نویسی

- **نسل اول: (زبان ماشین) (اواخر دهه ۱۹۴۰)**
  - استفاده از ارقام صفر و یک برای ایجاد کد دستورالعمل ها
- **نسل دوم: (زبان اسمبلی) (اوایل دهه ۱۹۵۰)**
  - استفاده از حروف برای کدگذاری کدهای ماشین (زبان نیمانیک)
- **نسل سوم: (زبان های سطح بالا) (اواخر دهه ۱۹۵۰)**
  - زبان های رویه گرا
  - اولین زبان فرترن
- **نسل چهارم: (زبان های فوق بالا) (اواسط دهه ۱۹۷۰)**
  - شبیه زبان های طبیعی
  - غیر رویه گرا؛ عدم تشریح چگونگی عملیات

دسته بندی سطح زبان های برنامه نویسی

سطح زبان:

- سطح پایین: (Macro-Assembler, Assembler)
  - دسترسی مستقیم به حافظه (زبان ماشین)
  - کار کردن با مفاهیم بیت، بایت و آدرس (با استفاده از مفهوم باینری)
- 0010 0000 0000 0100
- 1000 0000 0000 0101
- سرعت اجرای بسیار بالا
- سطح میانی: (Java, Forth, C, C++)
- سطح بالا: (Pascal, Ada, Cobol, Basic)
  - قابلیت خوانایی بالا (دستورالعمل ها به زبان محاوره ای نزدیک است)
- cost = price + tax;

نحوه و معنی (Syntax & Semantic)

- **نحوه:**
  - ساختار دستوری مربوط به جملات زبان نحوه نوشتن آن ها را مشخص می کند. (ظاهر زبان را Syntax گویند)
- **معنی:**
  - مفهومی که به آن ساختار منتسب می شود درک ما را از آن زبان و نحوه عملکرد زبان را نشان می دهد:
  - **مثال:**

- Pascal

```
Var A: array [0..9] of integer;
```

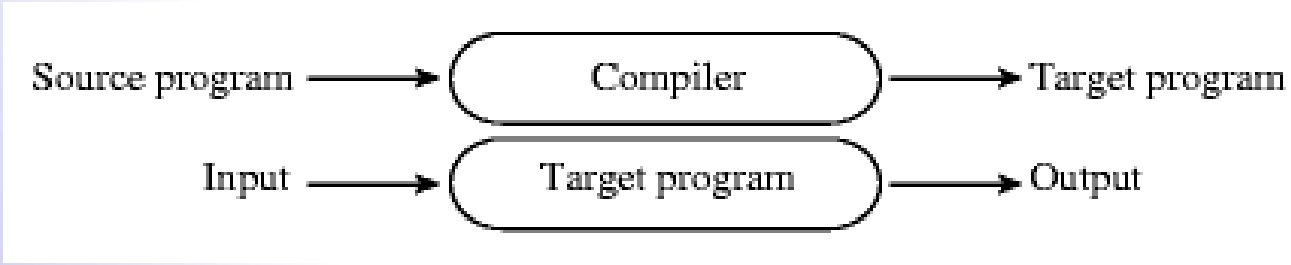
- C

```
int A[10];
```

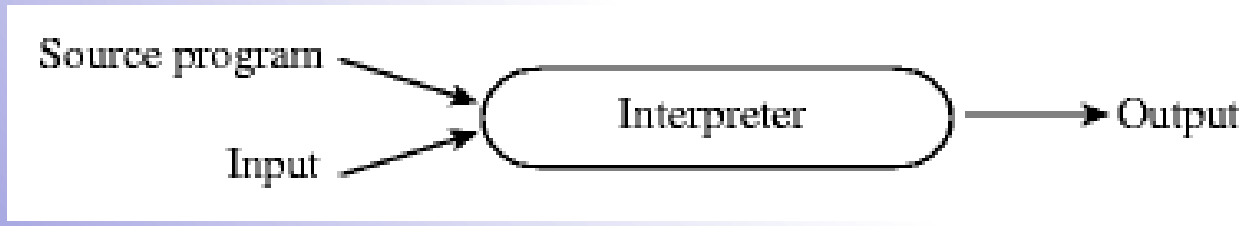
معنی تقریباً یکسان بازای دو نحوه نگارش متفاوت

اجرای برنامه

- **کامپایلر**
  - تغییر متن یک برنامه در زبان A بدون تغییر در معنی و مفهوم آن به زبان B
  - تبدیل زبان سطح بالا به زبان سطح پایین



- **مفسر**
  - برنامه ای برای دریافت یک برنامه با زبان منبع اجرای آن



دسته بندی ساختار زبان های برنامه نویسی

- ساختار زبان:

  - زبان ساخت یافته
    - Pascal, Ada, C, Java
    - قابلیت خوانایی و درک بالا
    - ساختار منظم متشکل از توابع
    - توانایی استفاده از حلقه های تکرار
    - زبانهای شیء گرا (Object Oriented) مانند C++
  - زبان غیر ساخت یافته
    - (Fortran, Cobol)

دلایل پیدایش برنامه نویسی ساخت یافته

- مشکلات برنامه های غیرساخت یافته
  - تأخیر در زمان بندی تولید نرم افزار
  - هزینه های بالای تولید
  - عدم وجود قابلیت اطمینان بالا

- ویژگی های برنامهٔ ساخت یافته
  - روش منظم برای نوشتن برنامه
    - تصور برنامه به صورت مجموعه ای از فعالیت ها بر روی داده ها
    - هر مسأله به مجموعه ای از مسائل کوچکتر تجزیه می شود.
    - تجزیه تا رسیدن به برنامه های ساده قابل درک ادامه می یابد.
  - خوانایی بالای برنامه
  - تست و اصلاح ساده

دلایل پیدایش برنامه نویسی شیء گرا

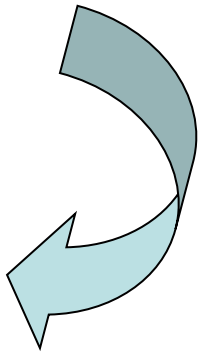
- امکان استفاده از قطعات نرم افزاری در برنامه های مختلف
  - افزایش سرعت تولید نرم افزار
  - افزایش قابلیت خوانایی برنامه
  - سادگی تست و اصلاح برنامه ها

هدف مدلسازی اشیای دنیای واقعی

- اشیاء (جاندار، بی جان)
- تمام اشیاء از طریق مشاهده و بررسی صفات و رفتارهای آنها قابل شناسایی هستند.

دلایل پیدایش برنامه نویسی شیء گرا - ۲

اشیاء ساخته شده در برنامه
به عنوان بخشی از برنامه ساخت یافته اند



باید با
برنامه نویسی ساخت یافته
به خوبی آشنا شد

موارد مهم در برنامه نویسی

- برنامه نویسی نوعی حل مسأله است
  - تعیین خواسته های مسأله
  - تحلیل مسأله
  - طراحی الگوریتم حل مسأله
  - پیاده سازی الگوریتم
  - تست و کنترل برنامه
  - نگهداری و نوسازی برنامه

اجزای حل مسأله

- تعیین خواسته های مسأله

  - شناخت مسأله به وضوح و بدون ابهام
  - درک موارد لازم برای حل مسأله
  - نادیده گرفتن جنبه های بی اهمیت و پرداختن به مسائل اصلی
  - در صورت لزوم مذاکره با طراح مسأله

اجزای حل مسأله – ۲

- تحلیل مسأله
  - تعیین ورودی ها و خروجی ها
    - ورودی: داده هایی که مسأله بر روی آنها کار می کند.
    - خروجی: نتایج مورد انتظار
  - تعیین نیازها و محدودیت های مسأله
  - تعیین فرمت اطلاعات خروجی و تعیین ارتباط ورودی و خروجی

اجزای حل مسأله – ۳

- طراحی الگوریتم
  - مشکل ترین بخش حل مسأله
  - نوشتن بخشها به صورت قدم به قدم برای حل مسأله (الگوریتم)
  - بررسی توانایی و صحت الگوریتم در حل مسأله
  - سعی در طراحی «بالا به پایین» در حل مسأله
  - مراحل کلی عمومی الگوریتم ها:
    - خواندن داده ها
    - انجام محاسبات
    - چاپ نتایج
  - حل هر مرحله به صورت جداگانه (بهینه سازی الگوریتم)

اجزای حل مسأله – ۴

- پیاده سازی الگوریتم
  - تبدیل الگوریتم به برنامه
  - تبدیل هر مرحله متناسب با برنامه به یک یا چند دستور زبان
- تست و کنترل برنامه
  - تعیین توانایی برنامه در اجرای خواستهٔ مورد نظر
  - اعمال ورودی های مختلف و بررسی حالات ممکن در مسأله
- نگهداری و نوسازی برنامه
  - اصلاح برنامه جهت حذف خطاهای قبلی
  - نوسازی آن جهت پاسخگویی به نیازهای فعلی