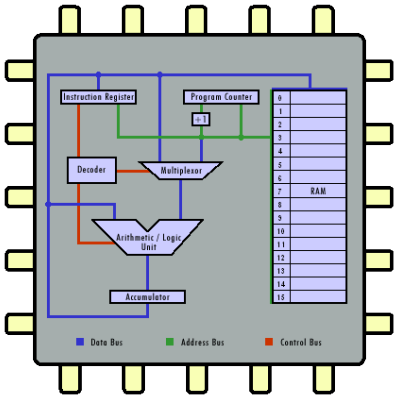
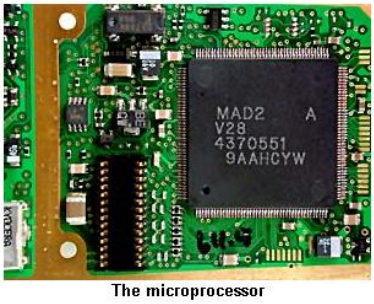
Primary Memory (or Main Memory)

- Stores program instructions and data needed for processing--the working memory of a computer (like a human's short-term memory).
- The primary storage area for **programs** and **data** in active use.
- Also know as RAM (Random Access Memory).
- It is volatile… it is erased when power is off.

CPU – The Central Processing Unit

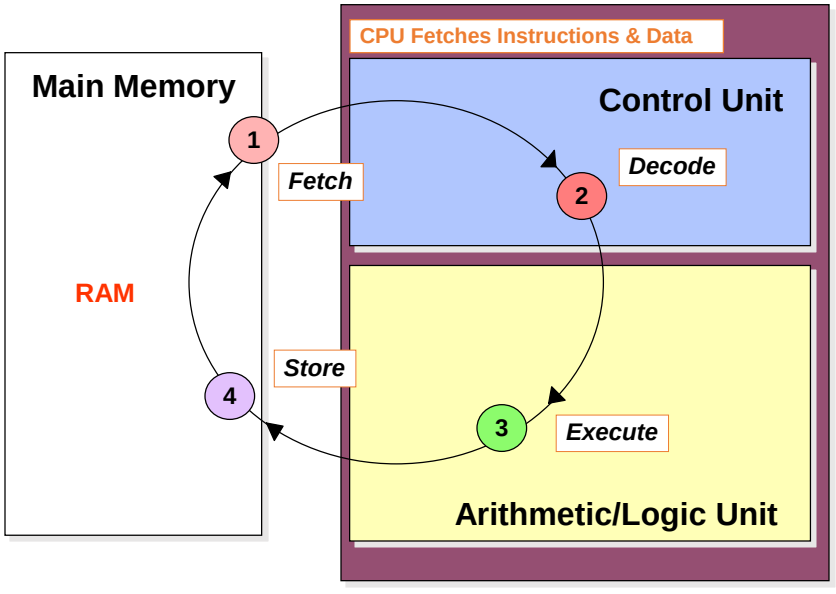
http://courses.cs.vt.edu/csonline/MachineArchitecture/Lessons/CPU/Lesson.html





The microprocessor

The Machine Cycle (aka Fetch-Execute Cycle)



Programming – Machine Languages

```
00000100 10000010
00000001 10000001
00000101 10000100
00001011 10000100
00001101 00010000
00010100 00000010
00000101 10000011
00001111 00000000
00010100 00000011
00000101 10000011
00001111 00000000
```

Programming – Assembly Languages

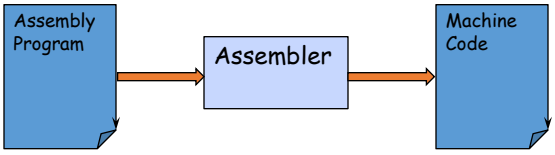
00000100 --> LOD (load)
00000001 --> SUB (subtract)
00000101 --> STO (store)
00000010 --> MUL (multiply)
10000001 --> X (memory location 129)
10000010 --> Y (memory location 130)
10000011 --> Z (memory location 131)
...

Programming – Assembly Languages

00000100 10000010	LOD Y
00000001 10000001	SUB X
00000101 10000100	STO T1
00001011 10000100	CPL T1
00001101 00010000	JMZ 16
00010100 00000010	LOD #2
00000101 10000011	STO Z
00001111 00000000	HLT
00010100 00000011	LOD #3
00000101 10000011	STO Z
00001111 00000000	HLT

Assembly Languages & Assemblers

00000100 --> LOD (load)
00000001 --> SUB (subtract)
00000101 --> STO (store)
00000010 --> MUL (multiply)
10000001 --> X (memory location 129)
10000010 --> Y (memory location 130)
10000011 --> Z (memory location 131)



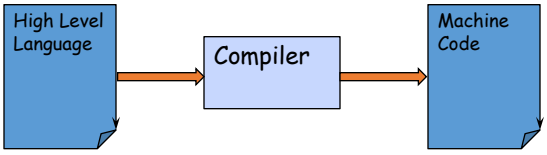
Programming – High Level Languages

```
LOD Y
SUB X
STO T1
CPL T1
JMZ 16
LOD #2
STO Z
HLT
LOD #3
STO Z
HLT
```

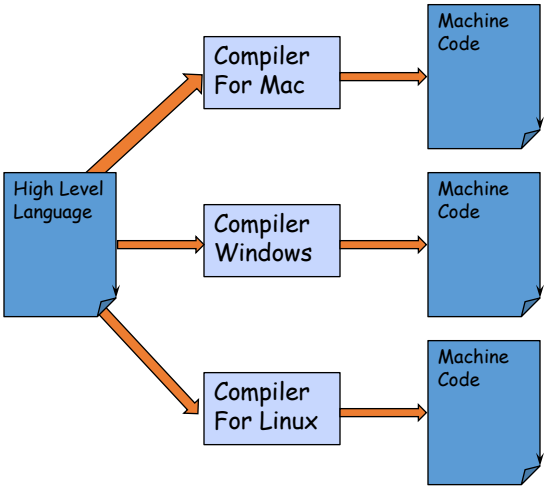
```
if (x > y) {
    z = 2;
} else {
    z = 3;
}
```

High Level Languages & Compilers

```
if (x > y) {
    z = 2;
} else {
    z = 3;
}
```

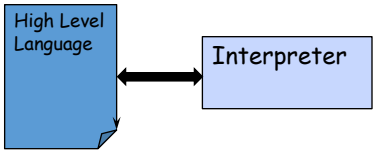


High Level Languages & Compilers



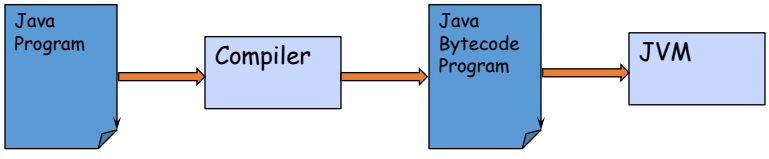
High Level Languages & Interpreters

```
if (x > y) {
    z = 2;
} else {
    z = 3;
}
```

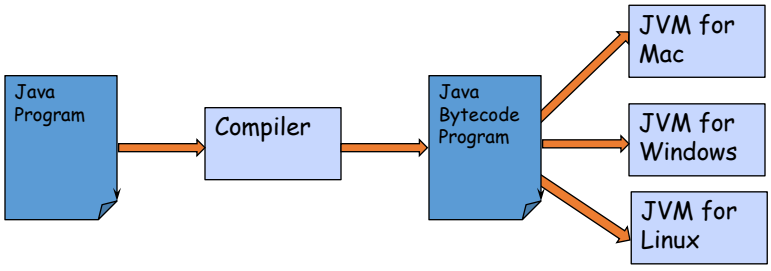


Java & the Java Virtual Machine (JVM)

```
if (x > y) {
    z = 2;
} else {
    z = 3;
}
```



The Java Virtual Machine (JVM) – Java and Java Bytecode



Java Byte Code

A Java **for loop**

```
for (int i = 2; i < 1000; i++) {
    for (int j = 2; j < i; j++) {
        if (i % j == 0)
            continue;
    }
    System.out.println(i);
}
```

A Java compiler might translate the Java code above into byte code as follows, assuming the above was put in a method:

```
0:  iconst_2
1:  istore_1
2:  iload_1
3:  sipush 1000
6:  if_icmpge 44
9:  iconst_2
10: istore_2
11: iload_2
12: iload_1
13: if_icmpge 31
16: iload_1
17: iload_2
18: irem
19: ifne 25
22: goto 38
25: iinc 2, 1
28: goto 11
31: getstatic #84; // Field java/lang/System.out:Ljava/io/PrintStream;
34: iload_1
35: invokevirtual #85; // Method java/io/PrintStream.println:(I)V
38: iinc 1, 1
41: goto 2
44: return
```