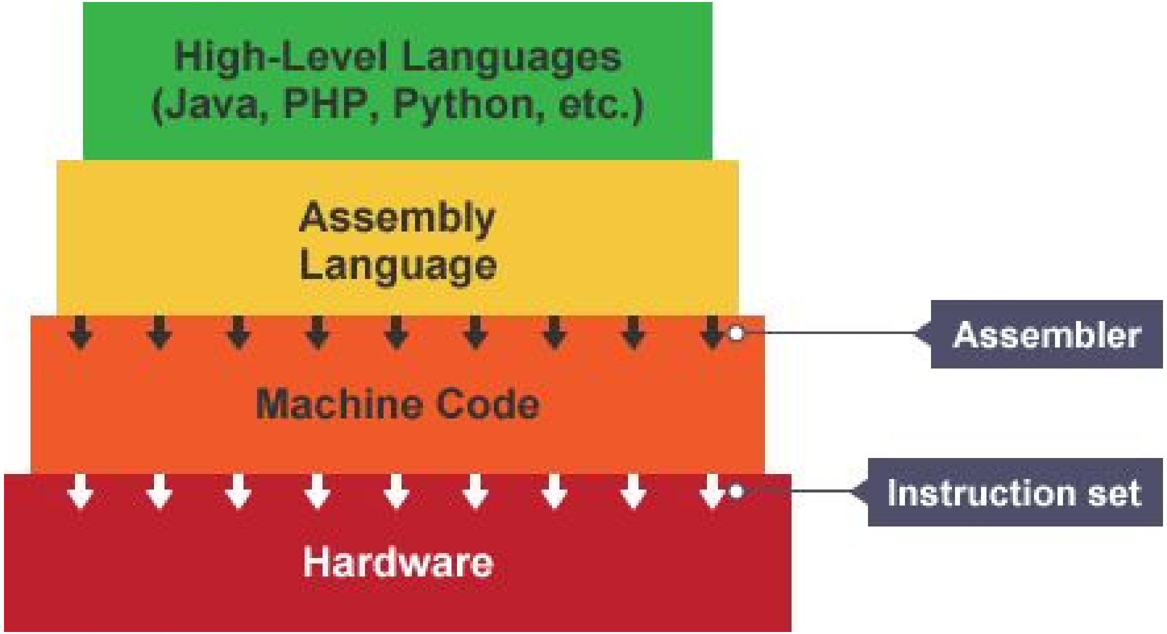
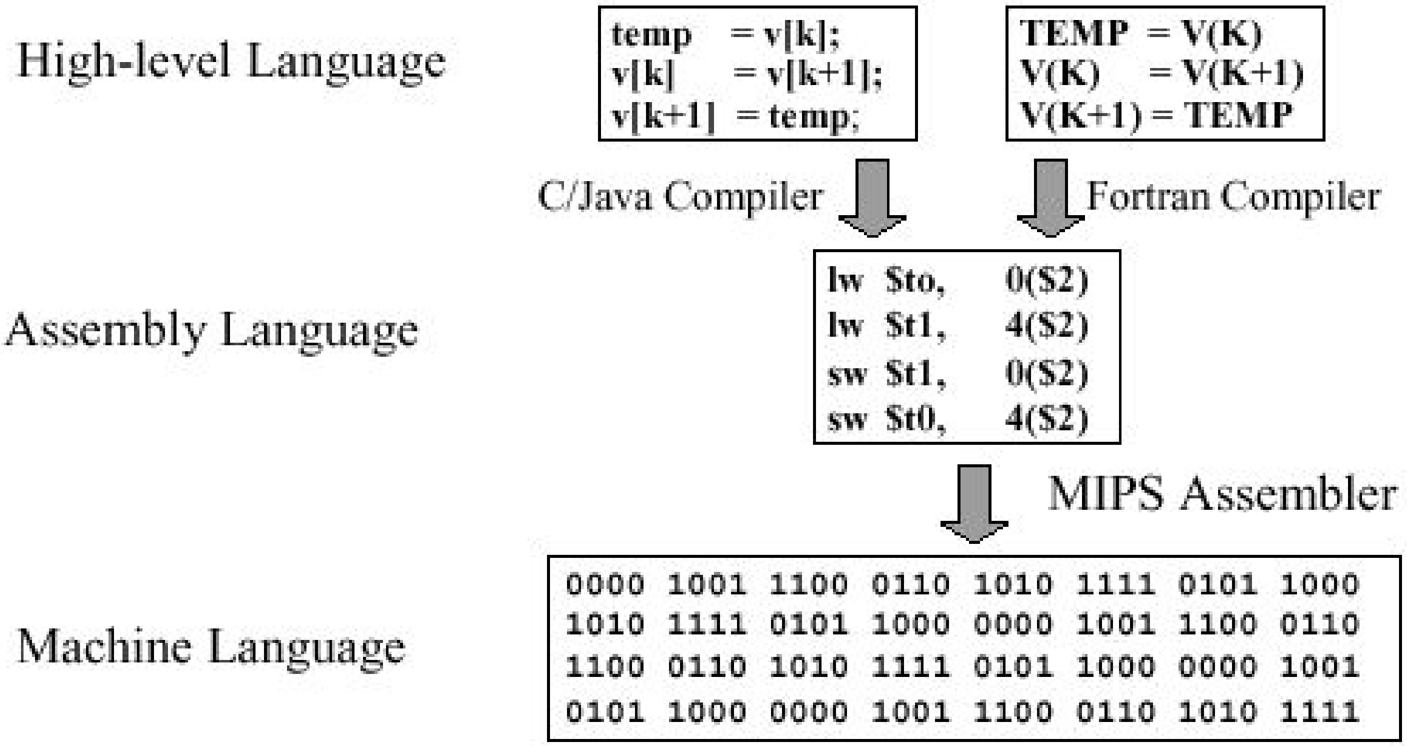
Reverse Engineering

Hack Night - Week 6

Code to executable



Assembly vs Machine Code



High-level Language

```
temp    = v[k];
v[k]    = v[k+1];
v[k+1]  = temp;
```

```
TEMP   = V(K)
V(K)   = V(K+1)
V(K+1) = TEMP
```

C/Java Compiler

Fortran Compiler

Assembly Language

```
lw  $to,   0($2)
lw  $t1,   4($2)
sw  $t1,   0($2)
sw  $t0,   4($2)
```

MIPS Assembler

Machine Language

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

Registers (read/write)

Registers are basically CPU variables that can hold the number of bits as the CPU architecture (32 bit processors can hold 32 bits of data in the registers)

Base pointer: ebp

Stack pointer: esp

Instruction pointer: eip

Temporary variables: eax, ebx, ecx (counter), edx (data), esi (source)

Flags (read only)

- Zero Flag
- Carry flag
- Overflow flag
- etc

ADD/SUB

ADD <dest>, <src>, n

dest = src + n

SUB <dest>, <src>, n

dest = src -n

INC

Increment value by one

```
inc <reg>
```

MOV

Moves data into dest from src

mov <reg>,<reg>

mov <reg>,<mem>

mov <mem>,<reg>

mov <reg>,<const>

mov <mem>,<const>

PUSH

Push onto top of stack

```
push <reg>
push <mem>
```

POP

Pop data off top of stack

pop <reg>

LEA

Load effective address

lea <reg>, <mem>

CMP

Compares two values and sets zero flag

cmp <reg>,<reg>

cmp <reg>,<mem>

cmp <mem>,<reg>

cmp <reg>,<con>

JMP/ JZ / JNZ / JE / JNE

Set EIP to new address based on zero flag

```
jmp <address>
jz <address>
jnz <address>
```

CALL/RET

Call function/subroutine or return

call <subroutine>

ret <val/none>