"Open the pod-bay doors, HAL."
Computer instruction from *2001: A Space Odyssey*

Machine Instructions

Computers do exactly what we tell them to do. The instructions are the means by which all tasks -- arithmetic, logic, input/output -- are expressed. Though most computers have a unique instruction set, the similarities are far greater than differences, justifying consideration of the MIPS instruction set.

© 1995 LS All Rights Reserved

Binary Review

- Whereas decimal represents numbers by their coefficients of powers of 10, binary expresses numbers by coefficients of powers of two:

$$
\begin{aligned}
&321_{10} \\
&= 3\times10^{2}+2\times10^{1}+1\times10^{0} \\
&= 256 + 64 + 1 \\
&= 1\times2^{8}+0\times2^{7}+1\times2^{6}+0\times2^{5}+0\times2^{4}+0\times2^{3}+0\times2^{2}+0\times2^{1}+1\times2^{0} \\
&= 1\ 0100\ 0001_{2}
\end{aligned}
$$

- Binary groups by 4 digits
- Hexadecimal uses base 16
- Convention: sequences of 0's and 1's should be considered to be binary.

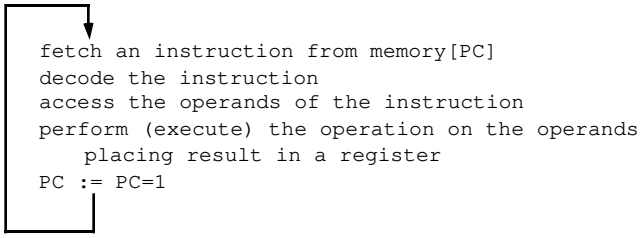
Binary Facts

$2^{10} \approx$ thousand
$2^{20} \approx$ million
$2^{30} \approx$ billion
$2^{40} \approx$ trillion
• Having a favorite binary number is not usually helpful in meeting people at a bar.

© 1995 LS All Rights Reserved

Instruction Interpretation

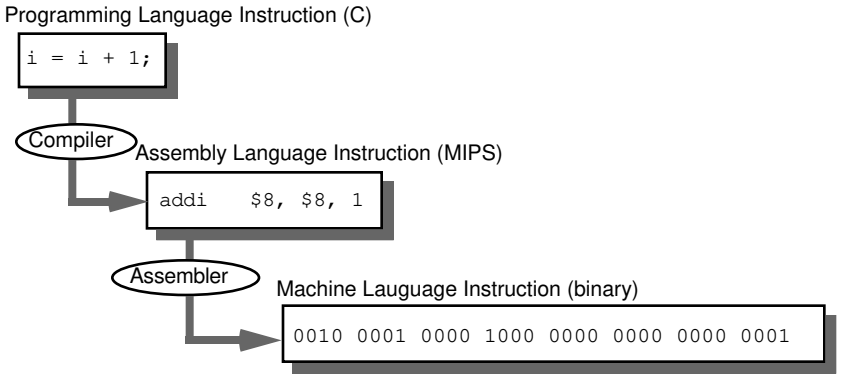
A computer is a hardware implementation of the "fetch/executed" cycle.



© 1995 LS All Rights Reserved

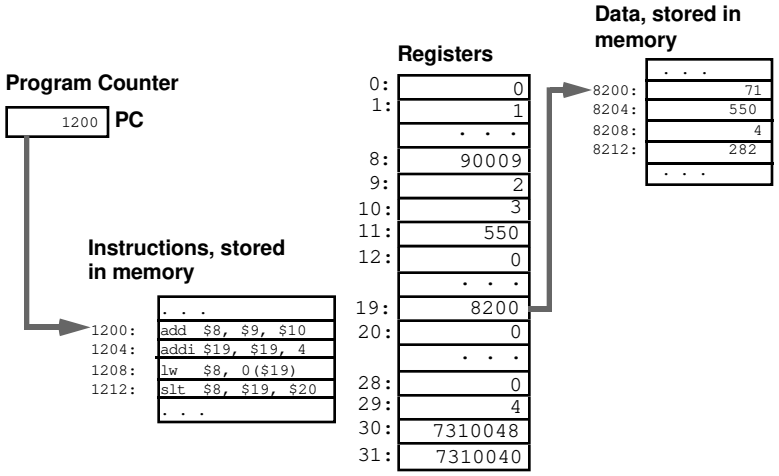
Changing Forms of Instructions

The instructions must have a form understandable to the follower. The computer only "understands" binary, so translators lower the level.



© 1995 LS All Rights Reserved

Components of Instruction Execution



© 1995 LS All Rights Reserved

Anatomy of an Instruction

```
add     $8, $9, $10      # $8 = $9 + $10

sub     $12, $12, $8     # $12 = $12 - $8
```

- Operation -- assembly language uses prefix form
- Comments follow #
- Instruction format (R) -- rigid operand type and order
  - Exactly 3 operands
  - Operands can only be register names
- Operands treated as signed integer values; there are separate instructions for floating point.

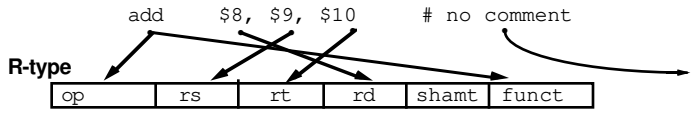
Reg7 = (Reg8+Reg9)-(Reg6+Reg5)
becomes . . .

```
add $10,  $8,  $9
add $11,  $6,  $5
sub  $7, $10, $11
```


© 1995 LS All Rights Reserved

Encoding R-type Instructions

- Instructions are 1 word long (32 bits)
- Partitioned into fixed length fields
  - op = operation (6)
  - rs = first source operand (5)
  - rt = second source operand (5)
  - rd = distination operand (5)
  - shamt = shift amount (5)
  - funct = function field (6)



R-type

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
0	9	10	8	0	32
000000	01001	01010	01000	00000	100000

sub has the same op=0, but funct=34

© 1995 LS All Rights Reserved

Immediate Instructions, I-type

Many operands are constant data: 52% (GCC), 69% (Spice)
Keep "small" constants in the instruction

```
addi    $10, $10, 1    # increment $10
```

I-type format

op	rs	rt	immediate
6 bits	5 bits	5 bits	16 bits
8	10	10	1
001000	01010	01010	0000 0000 0000 0001

What are the consequences?

© 1995 LS All Rights Reserved

Fact Sheet on Registers/Instructions

Machines typically have 16 or 32 registers

More registers => more processor storage

More registers => more instruction address bits

More registers => possibly slower clock speeds

Register $0 is the constant 0

Extending the instruction set . . .

```
sub $8, $0, $8        # negate  $8

add $9, $0, $5        # move $5, $9

addi   $10, $0, 4     # loadi $10, 4
```


© 1995 LS All Rights Reserved