

lecture 8

MIPS assembly language

non-executable
(ASCII file),
not machine specific

C, Fortran,
etc

+translation
needed

executable,
machine
specific

machine
code

C, Fortran

human
readable
machine
code
(ASCII)

01101011
00100101
...

assembly
language



machine code

compile

Java has another layer
(JVM - Java virtual Machine)

I'll explain the basic idea
at end of course,
time permitting.

COMP 273

human
readable
machine
code
(ASCII)

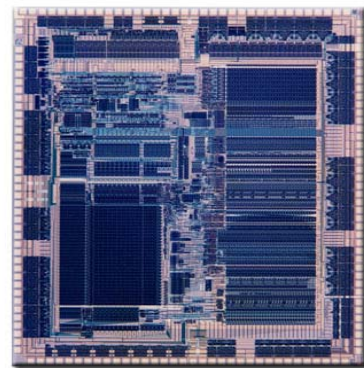
01101011
00100101
...

MIPS
assembly
language

MIPS
machine
code

assembler

MIPS R2000 CPU (1985)

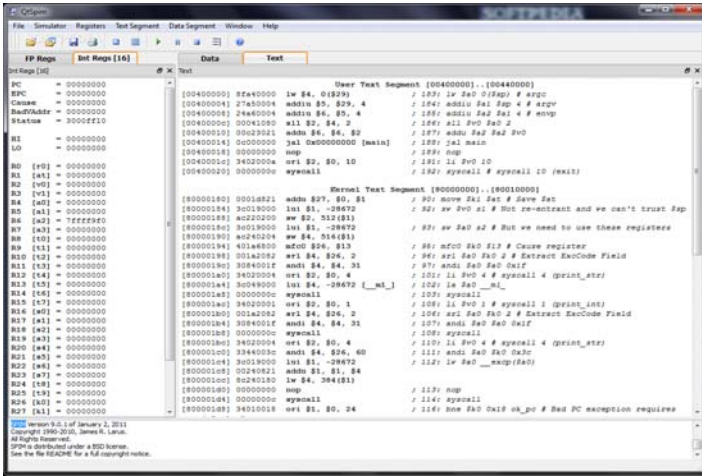


~ 1cm

- "Reduced Instruction Set Computer" (RISC)
- Keep it simple

SPIM (MIPS) simulator

runs on Windows, Mac, linux



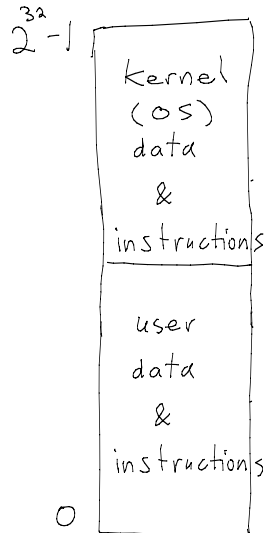
Addressing in MIPS

1 word = 4 bytes
= 32 bits



MIPS Memory

2^{32} bytes
= 2^{30} words



MIPS instructions

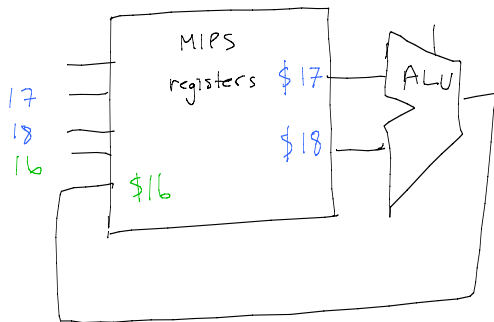
`add $16, $17, $18`

register 16 is assigned
the sum of registers
17 and 18

↑ comment symbol

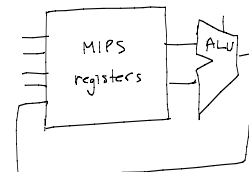
MIPS instructions

`add $16, $17, $18`



Arithmetic and logic instructions

`add $16, $17, $18`
`sub $19, $15, $19`
`and $17, $17, $16`
`or $16, $17, $18`
`nor $16, $17, $20`



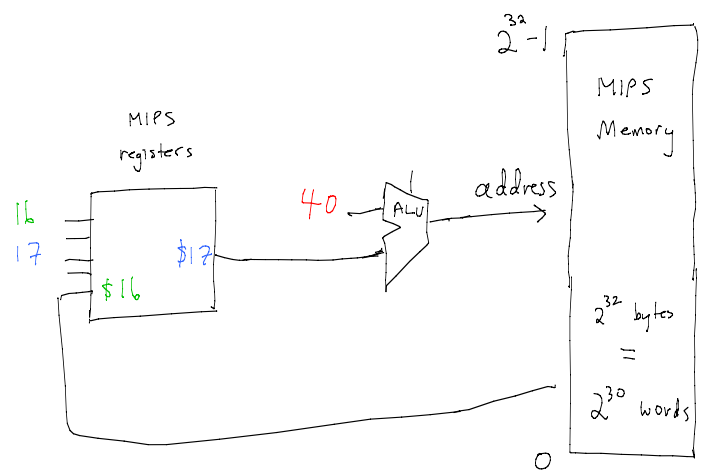
Memory transfer instructions

load word from Memory

lw \$16, 40(\$17)

↑ ↑ ↑
Copy word offset memory
from Memory address
address
\$17 + offset
to register 16

lw \$16, 40(\$17)



$X = Y + Z$
in registers in Memory

lw \$16, 40(\$17)

add \$18, \$20, \$16

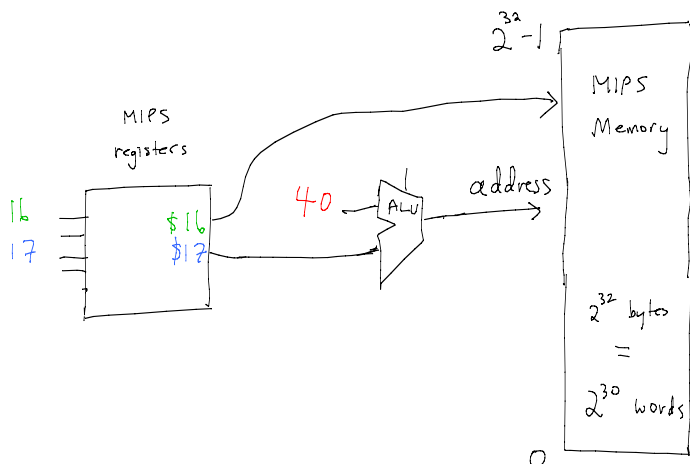
store word to Memory

sw \$16, 40(\$17)

Copy word from this register to memory at address specified by \$17 + offset

↑ ↑
offset memory
address

lw \$16, 40(\$17)



$Z = X + Y$
in Memory in registers

add \$18, \$16, \$20

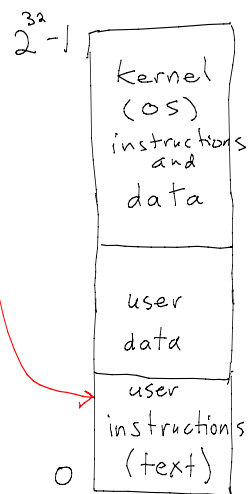
sw \$18, 40(\$17)

You now have a glimpse of MIPS instructions.

How do we go from one instruction to another?

Program Counter

- PC register specifies address of instruction that is currently being executed
- Default is to advance to next instruction.



Branching instructions

```
#
#      if (a != b)
#          f = g + h
#
```

```
beq $17, $18, Exit1
add $19, $20, $21
```

Exit 1 :

```
#      if (a != b)
#          f = g + h
#      else
#          f = g - h
#
```

```
beq $17, $18, Exit1
add $19, $20, $21
j   Exit2
```

```
Exit 1 : sub $19, $20, $21
Exit 2 :
```

human readable machine code (ASCII)

MIPS assembly language

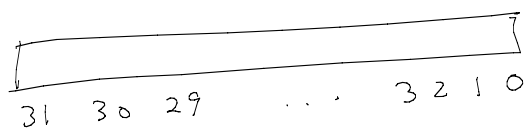
```
beq $17, $18, Exit1
add $19, $20, $18
Exit 1: sub $19, $20, $18
j   Exit2
Exit 2:
```

MIPS machine code

01101011
00100101
...

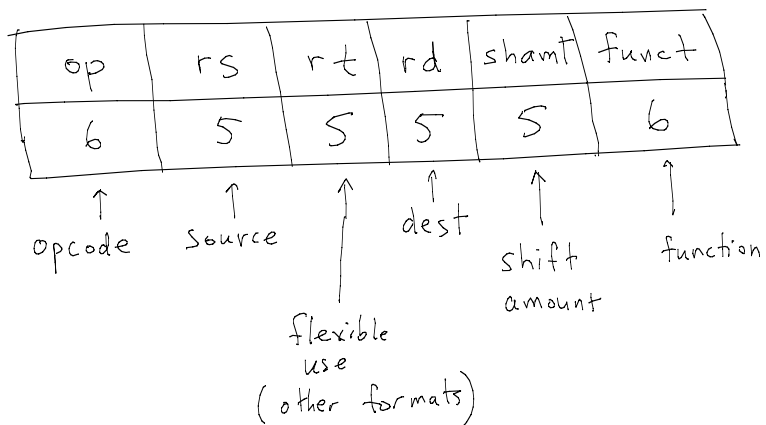


MIPS instructions (machine code)

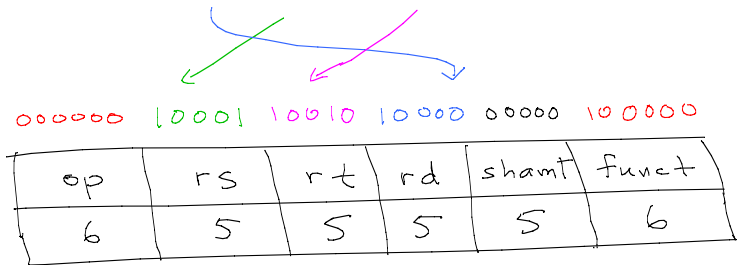


- always 32 bits (keep it simple)
- relatively few (RISC: reduced instruction set computer)
- three types: R, I, J

R format (R for register)

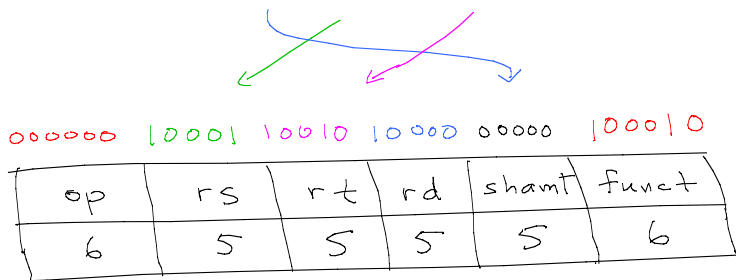


add \$16, \$17, \$18

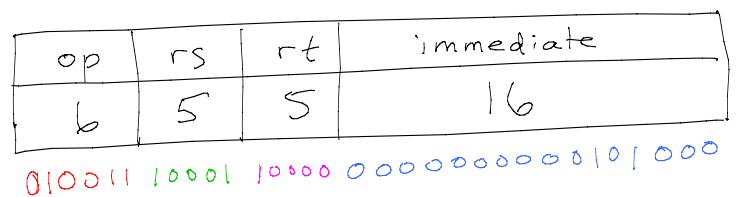


All R-format instructions have 000000 opcode.

sub \$16, \$17, \$18

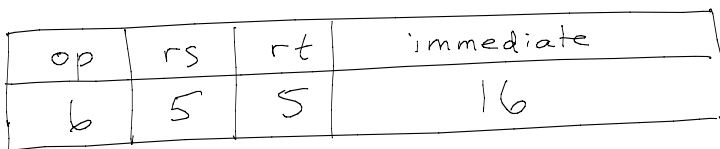


I format ("immediate")



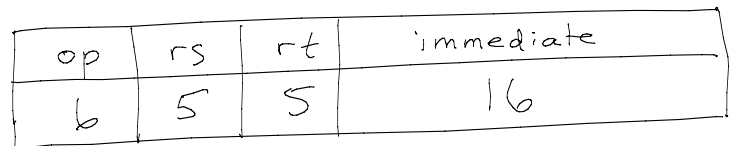
signed offset
from address in \$17

lw \$16, 40(\$17)



010011 10001 10000 0000000000101000

sw \$16, 40(\$17)

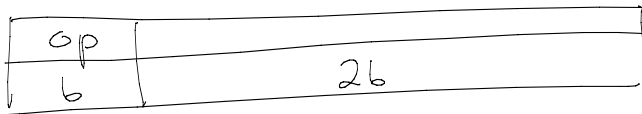


000100 10010 10001 offset

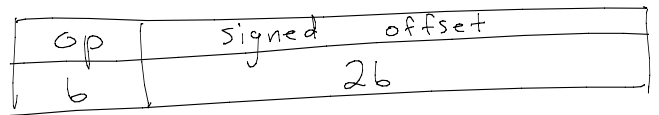
(number of words from
current instruction)

beq \$18, \$17, Exit1

J format (jump)



$$-2^{25} < \text{offset} < 2^{25} - 1$$



000010

(number of words from current instruction)

j Exit 2

Register names

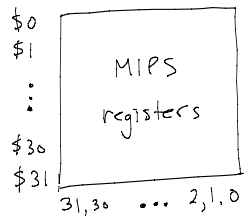
\$zero always has value 0.
(\$0)

"save"

\$s0, \$s1, ... \$s7
(\$16, \$17, ..., \$23)

"temporary"

\$t0, \$t1, ... \$t7
(\$8, \$9, ..., \$15)



We will learn other names later.

Why temporary?

$$g = g + h + i$$

add \$t0, \$s1, \$s2

add \$s1, \$t0, \$s3