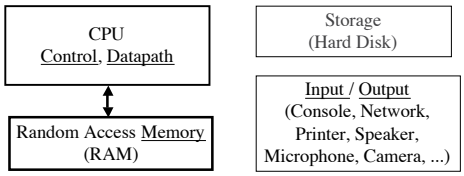
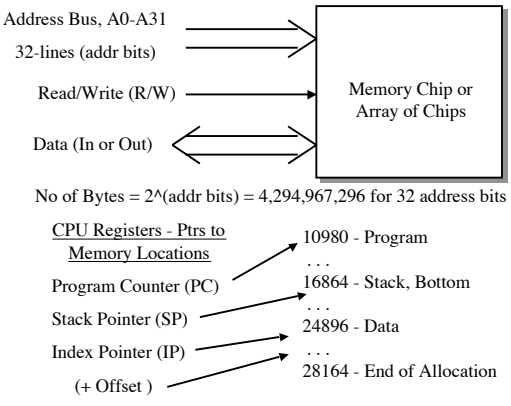
ECE 3055 Lecture 2

Instruction Sets

Reading: Comp. Org. & Design, Chap. 2, Sec.1-10, 19



The Five Classic Components of a Computer are Underlined



No of Bytes = 2^(addr bits) = 4,294,967,296 for 32 address bits

CPU Registers - Ptrs to Memory Locations

Program Counter (PC)

Stack Pointer (SP)

Index Pointer (IP)

(+ Offset)

10980 - Program
. . .
16864 - Stack, Bottom
. . .
24896 - Data
. . .
28164 - End of Allocation

2

Instruction Set Architecture - Must Match CPU Architecture

C code: X = Y + Z ;

Assembly (for MIPS):

```
lw $8, Y
lw $9, Z
add $10, $8, $9
sw $10, X
```

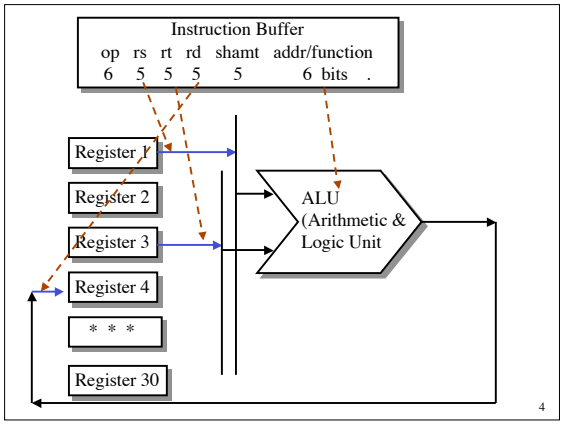
Machine Language (for MIPS, "add $10, $8, $9"):

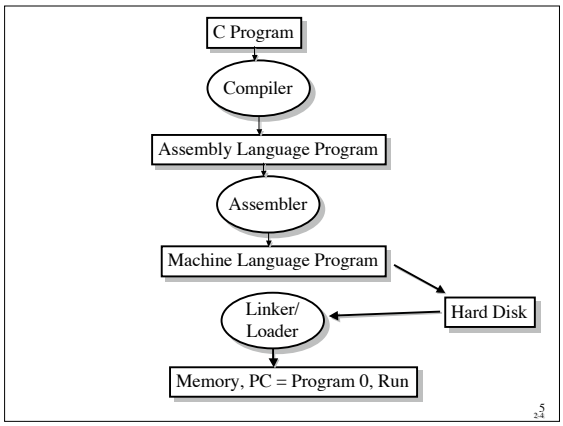
000000 01000 01001 01010 00000 100000

In decimal: op=0, rs=8, rt=9, rd=10, shamt=0, addr/funct=32

MIPS is a RISC. All machine instructions are 32-bits long.

3





The Power of Computing comes from Decisions (if's) and Iteration (loops and arrays)

C-code:

```
Loop: g = g + A[i] ;  //   add all A[i] values to g
      i = i + j ;
      if ( i != h) goto Loop ;
```

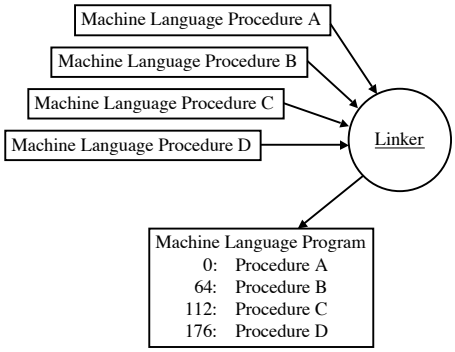
Assembly:

```
Loop: multi   $9, $19, $10    # $19 = i, $10 = 4
      lw      $8, Astart($9)  # $8 -> A (=&A[0])
      add     $17, $17, $8    # g = g + A[i]
      add     $19, $19, $20   # i = i + j
      bne     $19, $18, Loop  # branch on i != h
```

To branch on >, <, <= ,or >= requires two assembler instructions:

```
slt   $1, $16, $17   # $1 = 1 if $16 < $17, 0 if $16 >= $17
bne   $1, $0         # branch if $1 != 0 (if $16 < $17)
```

6 2-5



7

Procedure A calls a Procedure "B" (Subroutine B() in C):

A executes: "jal ProcedureAddress"

"jal" stores next instruction address (PC + 1) on $31

sets Instruction Addr Reg.(Program Counter) = ProcedureAddress

Procedure B starts running

must store value of $31 before it gets overwritten ("callee save")

must store values of any other registers before using them, except those to be used for return values.

does its job (e.g., I/O, sorts data in memory, puts Pi in $2, ...)

restores register values to the way A left them, including $31

executes: "jr $31"

"jr" resets IAR (PC) to value of $31

Procedure A then starts running again, exactly where it left off, except some registers have values returned from B, and/or data values in memory may have changed.

8

Other Addressing Modes

Immediate:

Constant value included in the last 16 bits of Instruction

Add 4 to $29

```
Assembler:   addi   $29, $29, 4  # add 4 to $29
Machine:
   op (6 bits) =8, rs (5 bits) = 26, rt (5 b) = 26, immed.(16b) = 4
   001000  11010  11010  11010  0000000000000100
```

Branch if $18 < 10

```
Assembler:   slti   $8, $18, 10    # set $8 to 1 if $18 < 10
             bne    $8, $0, Label  # branch to Label if $8 != 0
bne   $1, $0          # branch if $1 != 0 (if $16 < $17)
```

9

Unconditional Jump

Assembly: j 10000 # jump to location 10000

Machine: op (6 bits) = 2, address (26 bits) = 10000

000010 00000000000010011100010000

jump address is relative to PC+4 (the next normal instruction address)

note: 2^25-1 = +/- 33,554,431 - maximum jump distance

Conditional Jump

Assembly: beq $21, $8 Exit # jump to "Exit" if $21 == $8

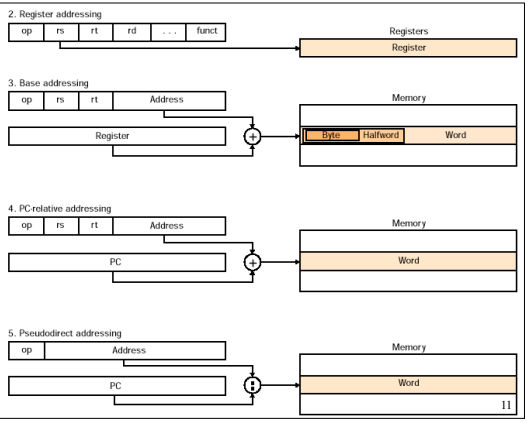
Machine: op (6 bits) = 5, rs (6) = 21, rt (6) = 8, address (16 b) = "Exit"

note: 2^15-1 = +/-32,767 - max. single-instruction jump distance.

Longer Jump: bnei $18, $19, 4 # jump over next instr. if $18 != $19

j Exit # now Exit can be 33MB distant.

10



11

MIPS is a "RISC"- Reduced Instruction Set Computer

All instructions are the same length (short).

Instructions execute quickly.

More instructions needed - larger memory required.

"CISC"- Classical Instruction Set Computer

Instructions vary in length (2, 4, 6, 8 bytes)

Instructions execution time varies, some are slow

Less instructions needed - smaller memory required

Example of instructions not in MIPS's Instruction Set:

Increment X (no INCR, no direct operation on memory locations

Increment, compare, and branch (if (++i < $12) goto Label)

Copy 1000 bytes from memory address X to memory address Y)

12