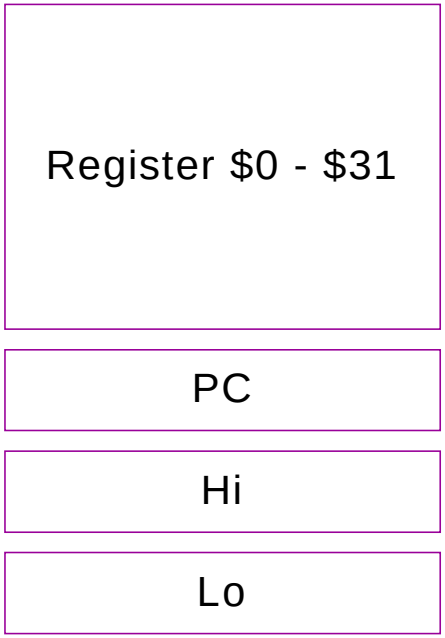
Chapter 2: Instructions: Language of the Machine

- Chapter overview:
  - MIPS instruction set architecture
    - instruction type
    - instruction format
    - addressing modes
  - Other instruction set architectures

MIPS R3000 Instruction Set Architecture: Registers

- Registers--32 general purpose registers
  - $zero (0) : 0
- 3 special purpose registers
    - PC: program counter
    - Hi, Lo for multiply and divide
- 2 remarks: register 2: $2 vs 2, $t2 vs $2
- Word length=32 bits



Memory Operands

- Memory is byte addressed
  - Each address identifies an 8-bit byte
- Words are aligned in memory
  - Address must be a multiple of 4
- MIPS is Big Endian
  - Most-significant byte at least address of a word
  - *c.f.* Little Endian: least-significant byte at least address

MIPS ISA: Instruction Categories

- Arithmetic & logic (AL)

  add $1, $2, $3 # $1 $2 + $3

  sub $1, $2, $3 # $1 ← $2 - $3

  – each AL inst. has exactly 3 operands

- Load/store -- data transfer instruction

  lw $1, x($2) # $1 = memory [$2+x]

  sw $1, x($2) # memory[$2+x]=$1

- Control purpose--Jump, branch, set on less than

  beq $2, $3, L1 #branch to f(L1) when ($2) = ($3)

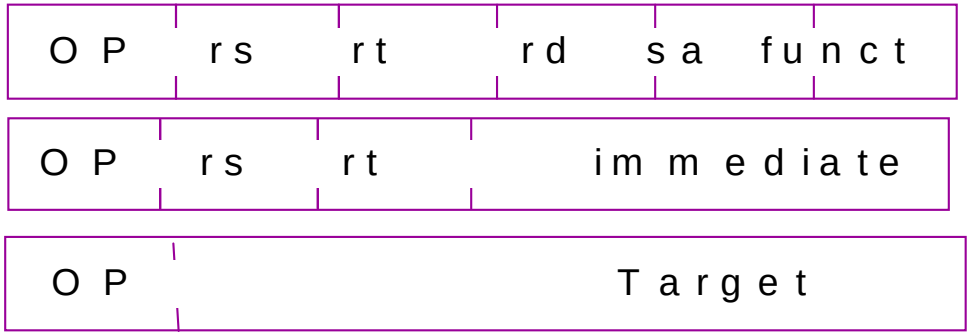
  bne $2, $3, L1 #branch to f(L1) when ($2) ($3)

  slt $2, $3, $4 #set on less than, $2=1 if ($3) < ($4)
  $2=0, otherwise

  j L1 # goto f(L1)

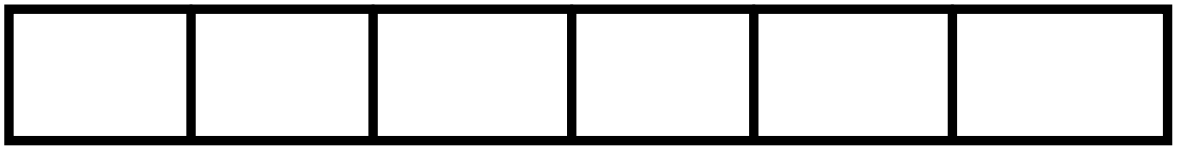
Representing Inst. in Computer

- Instruction format: layout of the instruction
- 3 types of formats:
  R-type (regular)
  I-type (Immediate)
  J-type (Jump)
- R-type: 6 fields
  - op: operation code
  - rs: 1st register source operand
  - rt: 2nd register source operand
  - rd: register destination
  - sa: shift amount
  - function: select the variant of operation in op field



Instruction Format/R-Type/Example

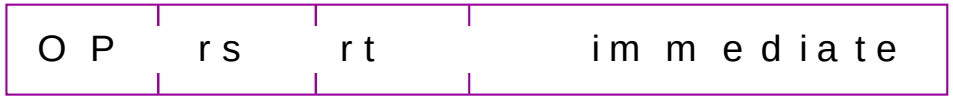
- Example: add $8, $17, $18



- Other R-type inst.
  sub $1, $2, $3
  slt $1, $2, $3
  jr $ra(31) #jump register, for returning the calling procedure

Instruction Format/I-type

- 4 fields, 32 bits



- Immediate (address) field: 16 bits, holds a constant
- Example: load word -- lw $8, Astart($19)

- Other I-type inst.
  add immediate -- addi $8, $9, 100
  branch on equal -- beq $1, $2, 100: goto *f*(100) if $1=$2

Instruction Format/J-Type

- 2 fields: 32 bits
- op: 6 bits
- address field: 26 bits
- Example: j 200 # go to location *f*(200)

- Other J type inst.:

jal 200 # jump & link, goto location *f*(200)

$31(ra)= PC + 4

MIPS Addressing Modes (p.116)

- Register addressing
  - operand is a register; e.g. add $8, $19, $18
- Base or displacement addressing
  - operand: at memory location reg. + constant (base)
    e.g. lw $8, 200($19)
- Immediate addressing
  - operand: constant, in inst.; e.g. addi $8, $8, 4
- PC-relative addressing
  - branch address = (PC=4) + (constant in inst.)*4
    e.g., bne $8, $21, Exit
- Pseudodirect addressing: jump address=(PC+4)(31-28)
  (concatenated with) jump constant*4

Basic ISA Classes

- Accumulator:
  - 1 address add A acc ← acc + mem[A]
- Stack (special purpose register):
  - 0 address add tos ← tos + next
- General purpose register:
  - 2 address add A, B loc(A) ← loc(A) + loc(B)
  - 3 address add A,B,C loc(A) ← loc(B) + loc(C)
    - memory/memory
    - memory/register
    - register/register --load/store (our MIPS)

Comparing Number of Inst.

Code sequence for C= A+B for 4 class of instruction sets:

Stack	Accumulator	GP Register Reg.-mem.	GP register load/store

Inst. count depends on the architecture

Evolution of Intel x86 (ch. 2.17)

- 1978: The Intel 8086 is announced (16 bit architecture): <100 instructions
- 1980: The 8087 floating point coprocessor is added
- 1982: The 80286 increases address space to 24 bits, +instructions
- 1985: The 80386 extends to 32 bits, new addressing modes
- 1989-1995: The 80486, Pentium, Pentium Pro add a few instructions (mostly designed for higher performance)
- 1997: 57 new "MMX" instructions are added, Pentium II
- 1999: The Pentium III added another 70 instructions (SSE)
- 2001: Another 144 instructions (SSE2)
- 2003: AMD extends the architecture to increase address space to 64 bits, widens all registers to 64 bits and other changes (AMD64)
- 2004: Intel capitulates and embraces AMD64 (calls it EM64T) and adds more media extensions
- ….
- 2008 … About 900 instructions
- "This history illustrates the impact of the "golden handcuffs" of backward compatibility

  "adding new features as someone might add clothing to a packed bag"

  "an architecture that is difficult to explain and impossible to love"

Fallacies

- Powerful instruction ⇒ higher performance
  - Fewer instructions required in the code
  - But complex instructions are hard to implement
    - May slow down all instructions, including simple ones
  - Compilers are good at making fast code from simple instructions

Example

- A processor with the following CPI. Suppose we can add new powerful Al instructions such that
  - Reduce 25% AL instructions
  - 10% increase of cycle time
- Is this a good choice?

Instruction	CPI	I. count
AL	1	500m
Load/store	10	300m
Branch	3	100m

Chapter Summary

- MIPS instruction set architecture
  - 3 categories: ALU, data transfer, branch and jump
  - 3 instruction types: R, I, J
  - 5 addressing modes: register, base, immediate, PC relative, pseudodirect
- Other ISA