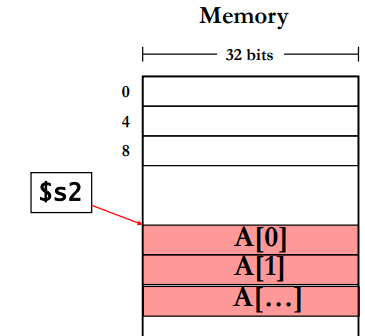
Computer Systems Lecture 5

Getting at the Data

Goal: g = h + A[0] where h is in register $s1, A[0] is the first element of array A and is pointed to by $s2

MIPS: lw $t1,0($s2)

Add $t2, $s1, $t1

Putting lw <register>,constant(<register>) tells the processor to get the data pointed to by the value in the specified register with an offset specified by the constant.  
Data-Transfer Instructions

Load Word: lw r1,n(r2) => r1 = memory[n+r2]

Store Word: sw r1,n(r2) => memory[2+r2]=r1

Load Byte: lb r1,n(r2) => r17-0 = memory[n+r2], r131-8 = sign extension

Store Byte: sb r1,n(r2) => memory[n+r2]=r17-0 no sign extension

Memory Addressing

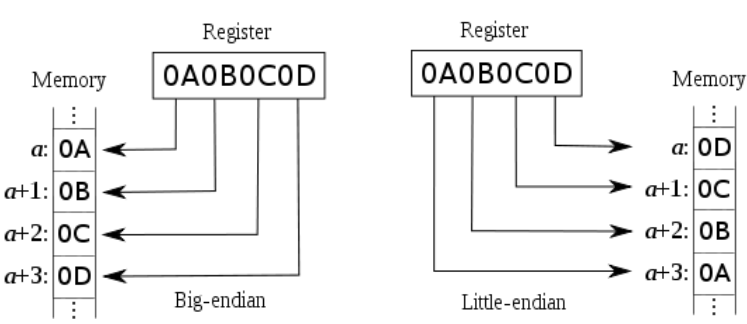
Memory is byte addressable, but it is organised so that a whole word can be accessed directly.

Where can a word be stored?

* Option 1: anywhere (unaligned)
* Option 2: at an address that is a multiple of the word size (aligned)
* Both options are used, MIPS requires alignment, x86 doesn’t

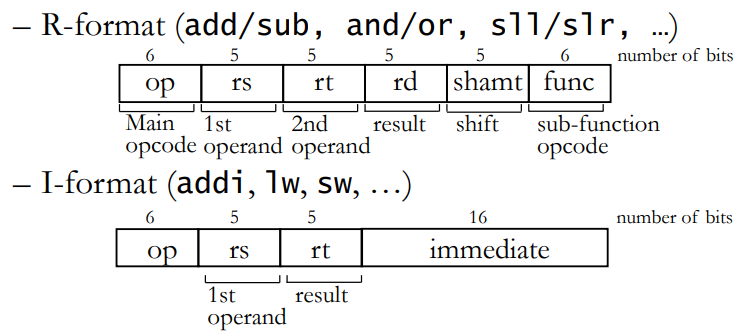
Memory Addressing: Endianness

Given a memory address, Endianness tells us where to find the starting byte of a word.



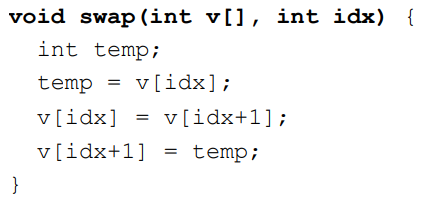
Instruction Formats

Instruction representation is composed of bit-fields, similar instructions have the same format. MIPS instruction formats are:

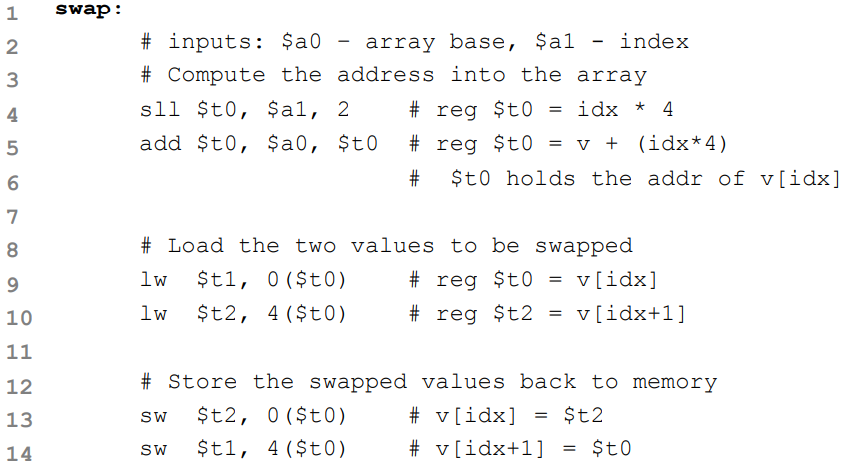
R formats are for working with registers

L formats are for working with memory

A Simple Function to Swap Array Elements

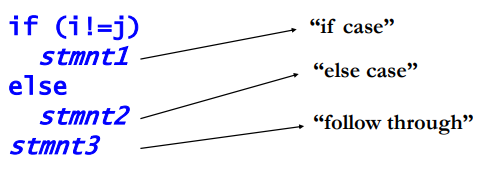


In MIPS this is written:

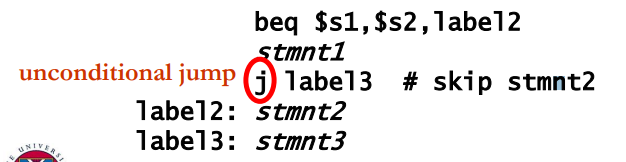


Control Transfers: If Structures

Java/C:

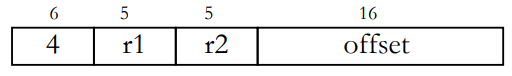


MIPS uses ‘branch if equal’: beq $s1, $s2, label. This compares the value in $s1 with the value in $s2, if they’re equal it will branch to the instruction marked label, so our earlier if would look like:



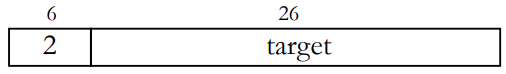
Control Transfer Instructions

Conditional branches, I-format: beq r1, r2, label



In assembly code, the label is usually a string, in machine code the lable is a PC-relative offset, the branch address is found using: branch target = PC + 4 \* offset. Similarly we have bne r1,r2,label (if r1 != r2 go to label (branch not equal)).

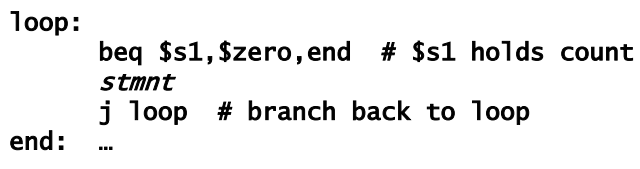
The unconditional jump has its own format know as J-format: j label



Loops in Assembly Language

Java/C: 

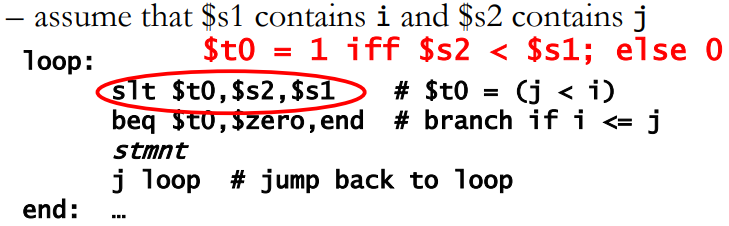
In MIPS this would be written:



Comparisons

In mips we have ‘set if less than’ (in R-format): slt r1 r1,r2,r3 => set r1 to 1 if r2 < r3, otherwise set r1 to 0

A use for this would be to build the equivalent of while (i > j) stmnt in MIPS:



Instruction Format Summary

* R-type
  + Three register operands
  + Most arithmetic, logical and shift instructions
* I-type
  + Instructions which use two registers and a constant
  + Arithmetic/logical with immediate operand
  + Load and store
  + Branch instructions with relative branch distance
* J-type
  + Jump instructison with a 26 bit address