Exercise classes computer architecture and system software

Yves Younan
DistriNet, Department of Computer Science
Katholieke Universiteit Leuven
Belgium
Yves.Younan@cs.kuleuven.ac.be

Overview

- Global overview: classes
- Registers
- Memory
- Instructions
  - Arithmetic instructions
  - Data transfer instructions
  - Logical instructions
  - Conditional/branch instructions
  - Unconditional jump instructions

Global overview

- Class 1 (today): Introduction, registers, simple operations, arrays
- Class 2: functions: stack, recursion
- Class 3 (computer): exercises on class 1 and 2 using SPIM
- Class 4: Pointers, addressing
- Class 5: Heap, linked lists
- Class 6 (Computer): exercises on class 4 and 5

Global overview

- Class 7: Macros, linkers and loaders
- Class 8: Data representation
- Class 9 (Computer): Exception handling, Input/ Output
- Class 10: Caches
- Class 11: Intel assembler
- Class 12 (Computer): Intel assembler, summary

Registers

- $t1-$t9: temporaries, $s0-$s7: saved registers
- $zero is always equal to 0
- $a0-$a3 used to pass arguments to functions
- $v0-$v1 used to return results
- $gp is the global pointer (class 5)
- $fp is the frame pointer (class 2)
- $sp is the stack pointer (class 2)
- $ra contains the return address (class 2)
- $at reserved for the assembler

Memory

- 2^30 memory words available
- A word is 4 bytes
- MIPS uses byte addressing so each word is 4 apart
- Instructions on words must be word aligned (i.e. multiple of 4)
- Memory holds information which can not be stored in registers

Instructions

- Most instructions can only use registers: data must be copied to registers from memory beforehand
- All instructions and their operands are 32 bits in size when encoded into machine language
- Results in 3 different types of instruction:
  - Registers
  - Immediates
  - Jump instructions
- Machine language instructions start with a 6 bit opcode, followed by operands

Instructions

- Registers in an instruction take up 5 bits (for registers 0-31)
- Immediate instructions:
  - Opcode 6 bits, register1 5 bits, register2 5 bits
  - Only 15 bits remaining for immediate
  - e.g.: addi $s1, $s2, 0x41414141 does not work, must do it in multiple times

Arithmetic instructions

- add $s1, $s2, $s3
  - $s1 = $s2 + $s3
  - Always takes three registers as operands
- addi $s1, $s2, 10
  - $s1 = $s2 + 10
  - Takes 2 registers and an immediate value as operands
  - Immediate value is maximum 16 bits long

Arithmetic instructions

- sub $s1, $s2, $s3
  - $s1 = $s2 - $s3
- No subi instruction
- mul $s1, $s2, $s3
  - $s1 = $s2 * $s3

Data transfer instructions

- Copy data from memory and back
- load word (copy from memory into register)
  - lw $s1, 100($s2)
  - $s1 = Memory[$s2 + 100]
  - $s2 contains the memory address, 100 is the offset from that address
- store word (save register in memory)
  - sw $s1, 100($s2)
  - Memory[$s2 + 100] = $s1

Logical instructions

- and $s1, $s2, $s3
  - $s1 = $s2 & $s3
- or $s1, $s2, $s3
  - $s1 = $s2 | $s3
- nor $s1, $s2, $s3
  - $s1 = ~($s2 | $s3)
- andi $s1, $s2, 100
  - $s1 = $s2 & 100

Logical instructions

- ori $s1, $s2, 100
- sll $s1, $s2, 10
  - $s1 = $s2 << 10
  - Shift left by X bits (same as multiply by 2^X)
- srl $s1, $2, 10
  - $s1 = $s2 >> 10
  - Shift right by X bits

Conditional/branch instructions

- beq $s1, $s2, Label
  - Branch equal
  - if ($s1 == $s2) goto Label
- bne $s1, $s2, Label
  - Branch not equal
  - if ($s1 != $s2) goto Label

Conditional/branch instructions

- slti $s1, $s2, 100
  - Set less than immediate
  - if ($s2 < 100) $s1 = 1 else $s1 = 0
- slt $s1, $s2, $s3
  - Set less than
  - if ($s1 < $s3) $s1 = 1 else $s1 = 0

Unconditional jump instructions

- j Label
  - goto Label
- jr $ra
  - goto $ra
  - jump execution to address stored in $ra