Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Archana Chidanandan and Larry Merkle

Homework 2 - Solutions
Instruction Formats and Addressing Modes
Max Points: 45 points

Directions

This assignment is due Tuesday, September 27 for all three sections. Submit your solutions on a separate sheet of paper. You may use SPIM and a calculator as required.

Learning Objectives

In the process of completing this homework assignment, students will develop their abilities to

- Explain the binary representation of various forms of data, including opcodes, registers, immediate values, and branch and jump target addresses.

Problems

1. [25 pts] Consider the MIPS code fragment on the following page (which doesn't do anything of any obvious practical value). Parts of several machine language instructions have been replaced by question marks. For each of those instructions except the first (and only those instructions), provide the following information:

   a. Address (assume that the address of the first instruction is 0x0040ffd8)
   b. Instruction format
   c. Name of each field and its value in either decimal or hexadecimal representation
   d. Addressing mode of each operand
   e. Complete machine language instruction in hexadecimal representation

   For example, for the first instruction

   a. The address is 0x0040ffd8.
   b. The format is I-type.
   c. The fields are opcode = 15, rs = 0, rt = 1, and immediate = 0x0fff.
   d. The addressing modes of $0, $1, and 0x0fff are register, register, and immediate, respectively.
   e. The complete machine language instruction is 0x3c010fff.

Rose-Hulman Institute of Technology
Computer Science and Software Engineering
Profs. Archana Chidanandan and Larry Merkle

Line	Assembly Language Instruction	Machine Language Instruction(s)	Address
1	li $t6, 0xfffffec	0x3c01????	0x0040ffd8
		0x342e????	0x0040ffdc
2	move $t0, $a0	0x00044021	0x0040ffe0
3	move $t1, $a1	0x00054821	0x0040ffe4
4	li $t4, 0	0x340c0000	0x0040ffe8
5	add $t4, $t4, 1	0x218c????	0x0040ffec
6	lw $t3, 0($t0)	0x8d0b0000	0x0040fff0
7	slt $t5, $t3, $zero	0x0160682a	0x0040fff4
8	LblTwo: beq $t3, $zero, LblOne	0x1160????	0x0040fff8
9	add $11, $11, $12	0x????????	0x0040fffc
10	sw $t3, -4($t1)	0xad2b????	0x00410000
11	add $t1, $t1, 4	0x21290004	0x00410004
12	j LblTwo	0x????????	0x00410008
13	LblOne:		0x0041000c

Instruction 1b. 0x342e????

a. The address is 0x0040ffdc.
b. The format is I-type.
c. The fields are opcode = 13, rs = 0, rt = 12, and immediate = 0xffec.
d. The addressing modes of $0, $13, and 0x0ffec are register, register, and immediate, respectively.
e. The complete machine language instruction is 0x342effec.

Instruction 5 0x218c????

a. The address is 0x0040ffec.
b. The instruction translates to an "addi" and the format is I-type.
c. The fields are opcode = 8, rs = 12, rt = 12, and immediate = 1.
d. The addressing modes of $12 and 1 are register and immediate respectively.
e. The complete machine language instruction is 0x218c0001.

Instruction 8 0x1160????

a. The address is 0x0040fff8.
b. The format is I-type.
c. The fields are opcode = 4, rs = 11, rt = 0, 16-bit immediate = 4 *(Note: For the 16-bit immediate field, SPIM gives the count of instructions from the current instruction, and not the PC.)*
d. The addressing modes of $11, $0 and the 16-bit immediate 0x0004 are register, register and PC-relative addressing respectively.
e. The complete machine language instruction is 0x11600004.

Instruction 9. 0x????????

a. The address is 0x0040fffc
b. The format is R-type.
c. The fields are opcode = 0, funct = 32; rs = 11, rt = 12, rd = 11.
d. The addressing mode for all three operands is register.
e. The complete machine language instruction is 0x016c5820.

Instruction 10. 0x`0xad2b????`

a. The address is 0x00410000
b. The format is I-type.
c. The fields are opcode = 32, rs = 9, rt = 11, 16-bit immediate = -4 = 0xfffc
d. The addressing mode for $9, $11 is register and for -4 is immediate.
e. The complete machine language instruction is 0xad2bfffc.

Instruction 12. `0x????????`

a. The address is 0x00410008
b. The format is J-type.
c. The fields are opcode = 2, 26-bit immediate = 0x0103ffe
d. The addressing mode for the 26-bit immediate field is pseudo-direct.
e. The complete machine language instruction is 0x08103ffe.

2. [20 pts] Exercise 2.52 (on the companion CD, follow the "In More Depth" link)

Accumulator		
Instruction	**Code bytes**	**Data bytes**
`load b  # Acc = b;`	3	4
`add c  # Acc += c;`	3	4
`store a  # a = Acc;`	3	4
`add c  # Acc += c;`	3	4
`store b  # Acc = b;`	3	4
`neg  # Acc =- Acc;`	1	0
`add a  # Acc -= b;`	3	4
`store d  # d = Acc;`	3	4
Total:	22	28

Code size is 22 bytes, and memory bandwidth is 22 + 28 = 50 bytes.

Stack		
Instruction	**Code bytes**	**Data bytes**
`push b`	3	4
`push c`	3	4
`add`	1	0
`dup`	1	0
`pop a`	3	4
`push c`	3	4
`add`	1	0
`dup`	1	0
`pop b`	3	4
`neg`	1	0
`push a`	3	4
`add`	1	0
`pop d`	3	4
Total:	27	28

Code size is 27 bytes, and memory bandwidth is 27 + 28 = 55 bytes.

Memory-Memory		
Instruction	**Code bytes**	**Data bytes**
`add a, b, c # a=b+c`	7	12
`add b, a, c # b=a+c`	7	12
`sub d, a, b # d=a-b`	7	12
Total:	21	36

Code size is 21 bytes, and memory bandwidth is 21 + 36 = 57 bytes.

Load-Store		
Instruction	**Code bytes**	**Data bytes**
`load  $1, b  # $1 = b;`	4	4
`load  $2, c  # $2 = c;`	4	4
`add  $3, $1, $2  # $3 = $1 + $2`	3	0
`store $3, a  # a = $3;`	4	4
`add  $1, $2, $3  # $1 = $2 + $3;`	3	0
`store $1, b  # b = $1;`	4	4
`sub  $4, $3, $1  # $4 = $3 - $1;`	3	0
`store $4, d  # d = $4;`	4	4
Total:	29	20

Code size is 29 bytes, and memory bandwidth is 29 + 20 = 49 bytes.

The load-store machine has the lowest amount of data traffic. It has enough registers that it only needs to read and write each memory location once. On the other hand, since all ALU operations must be separate from loads and stores, and all operations must specify three registers or one register and one address, the load-store has the worst code size. The memory-memory machine, on the other hand, is at the other extreme. It has the fewest instructions (though also the largest number of bytes per instruction) and the largest number of data accesses.