Hacettepe University
Computer Engineering Department
BBM234 Computer Organization
2019-2020 Spring Term

Homework 1 Solutions

Assigned date : 06.04.2020
Due date : 13.04.2020 through submit.cs.hacettepe.edu.tr as a single PDF file.

Questions: (Each one is 25 points.)

Q1. MIPS architecture has some conditional branches and unconditional jumps. We list some of them below. For each instruction type, write the maximum number of instructions between the current program counter (PC) and the target instruction. You should also write the instruction type.

Instruction	Maximum number of instructions that we can jump over	Instruction type
J	2^{26}	**J**
JR	2^{30}	**R**
JAL	2^{26}	**J**
BEQ	2^{15}	**I**
BNE	2^{15}	**I**

Q2. Write the 32-bit machine codes for the MIPS instructions given below. The opcode and function field of each instruction is given in the same line.
First, you should show the instruction format and the content of each field. Then, write the hexadecimal value to the table below.

Address		Instruction		
0x40000000	L1:	add $7, $7, $8	# funct:	add = 0x20
0x40000004		addi $7, $9, -3	# opcode:	addi = 0x08
0x40000008		.		
		.		
0x40000010		bne $6, $7, L1	# opcode:	bne = 0x05
0x40000014		jal func	# opcode:	jal = 0x03
		...		
0x4000002C	func:	…		

Solution:

add $7, $7, $8 Syntax: add $d, $s, $t

opcode	rs	rt	rd	shamt	funct
000000	00**111**	**0**1000	**0011**1	000**00**	**10**0000

0 0 E 8 3 8 2 0

addi $7, $9, -3 Syntax: addi $t, $s, imm

opcode	rs	rt	immediate
001000	01**001**	**0**0111	**1111** 1111 **1111** 1101

2 1 **2** 7 **F** F **F** D

bne $6, $7, L1 Syntax: bne $rs, $rt, offset

opcode	rs	rt	immediate
000101	00**110**	**0**0111	**1111** 1111 **1111** 1011

1 4 **C** 7 **F** F **F** B

Immediate value is -5. Since PC has already been updated with the address of the next instruction (0x40000014), branch will go back to L1 (PC + (imm << 2)).

jal func Syntax: jal target

opcode	address
000011	00 **0000** 0000 **0000** 0000 **0000** 1011

0 C **0** 0 **0** 0 **0** B

To compute the addres part of the JAL instruction, we discard 2 least and 4 most significant bits of the func address. The remaining 26 bits are written as the address part of the instruction.

Instruction	Hexadecimal value
add $7, $7, $8	**0x00E8_3820**
andi $7, $9, -3	**0x2127_FFFD**
bne $6, $7, L1	**0x14C7_FFFB** (0x14E6_FFFB will also be accepted, although incorrect)
jal func	**0x0C00_000B**

Q3. a) Write the values of the registers after the following MIPS program finishes its execution.

```
        lui $s0, 0x1234
        ori $s0, $s0, 0x0335
        andi $s0, $s0, 0x000F
        sra $s1, $s0, 2
        or $s2, $s0, $s1
        slt $s3, $s1, $s2
        bne  $s1, $s3, else
        addi $s2, $s2, -1
else:   sll $s4, $s2, 2
        jr $ra
```

s0	**s1**	**s2**	**s3**	**s4**
5	**1**	**4**	**1**	**16**

b) For the given "*number*" value, what does function f1 do? Write output values (value in s0) for the given *number* values in the table.

```
main:   addi $a0, $0, number
        addi $sp, $sp, -4
        sw $ra, 0($sp)
        jal f1
        add $s0, $v0, $0
        lw $ra, 0($sp)
        addi $sp, $sp, 4
        jr $ra    #exit

f1:     addi $t0, $0, 0
        addi $v0, $0, 1
        bne $a0, $0, else
        jr $ra
else:   beq $a0, $t0, done
        addi $t0, $t0, 1
        mul $v0, $v0, $t0
        mflo $v0
        j else
done:   jr $ra
```

Write the description of f1 below:

F1 calculates the factorial of a given number.

Number	0	3	5
S0	**1**	**6**	**120**

Q4. You have four instructions stored in the memory as given in the following table:

Instructions	Address	Instruction
Inst1	0x00400000	0x14100003
Inst2	0x00400004	0x012A4025
Inst3	0x00400008	0x2210FFFB
Inst4	0x0040000C	0x08100000
Inst5	0x00400010	---

a) Write the binary values for each instruction. Clearly show which bits corresponds to which field in the instruction format (opcode, rs, rt, rd, etc.).

Instructions	Instruction format
	BNE -- Branch on not equal, syntax: bne $s, $t, offset - encoding: 0001 01ss ssst tttt iiii iiii iiii iiii
0x14100003	**0001 01 | 00 000 | 1 0000 | 0000 0000 0000 0011**
	OR -- Bitwise or, syntax: or $d, $s, $t - encoding: 0000 00ss ssst tttt dddd d000 0010 0101
0x012A4025	**0000 00 | 01 001 | 0 1010 | 0100 0 | 000 00 | 10 0101**
	ADDI -- Add immediate, syntax: addi $t, $s, imm - encoding: 0010 00ss ssst tttt iiii iiii iiii iiii
0x2210FFFB	**0010 00 | 10 000 | 1 0000 | 1111 1111 1111 1011**
	J -- Jump, syntax: j target - encoding: 0000 10ii iiii iiii iiii iiii iiii iiii
0x08100000	**0000 10 | 00 0001 0000 0000 0000 0000 0000**

b) Write down the corresponding MIPS assembly code below for each machine code.

Instructions	MIPS Code
Inst1	**Label: bne $0, $s0, Done**
Inst2	**or $t0, $t1, $t2**
Inst3	**addi $s0, $s0, -5**
Inst4	**j Label**
	Done:

Name	**Register**
$0	0
$at	1
$v0-$v1	2-3
$a0-$a3	4-7
$t0-$t7	8-15
$s0-$s7	16-23
$t8-$t9	24-25
$k0-$k1	26-27
$gp	28
$sp	29
$fp	30
$ra	31

Instruction	**Opcode**
j	000010
jal	000011
beq	000100
bne	000101
addi	001000
slti	001010
andi	001100
ori	001101
xori	001110
lui	001111
lw	100011
sw	101011

Instruction	**Funct**
sll	000000
srl	000010
sra	000011
jr	001000
div	011010
add	100000
sub	100010
and	100100
or	100101
xor	100110
nor	100111
slt	101011