Hacettepe University
Computer Engineering Department
BBM234 Computer Organization
2017-2018 Spring Term

Homework 2

Assigned date : 13.3.2018
Due date : 20 or 22 March, 2018

You must hand in your homework in class. It does not matter in which day you submit your homework, either 20th or 22nd of March. Don't email your homework!
LATE HOMEWORKS WILL NOT BE ACCEPTED…

Questions: (Each one is 20 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		
JR		
JAL		
BEQ		
BNE		

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:    andi = 0x08
0x40000008           .
                     .
0x40000010           bne $6, $7, L1         # opcode:    bne = 0x05
0x40000014           jal func               # opcode:    jal = 0x03
                     ...
0x4000002C    func:  …
```

Instruction	Hexadecimal value
add $7, $7, $8	
andi $7, $9, -3	
bne $6, $7, L1	
jal func	

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**

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:

Number	0	3	5
S0			

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
0x14100003	
0x012A4025	
0x2210FFFB	
0x08100000	

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

Instructions	MIPS Code
Inst1	
Inst2	
Inst3	
Inst4	

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

Q5. a) Write the values of the registers and the stack for the following MIPS program. The value of the stack pointer is initially sp=0x7FFFFFFC.

Address		**Instructions**			
0x00400000		lui $s0, 0x1000		s0=	
0x00400004		ori $s0, $s0, 0x0008		s0=	
0x00400008		lw $a0, -4($s0)		a0=	
0x0040000C		addi $a1, $s0, -8		a1=	
0x00400010		lw $s1, 4($s0)		s1=	
0x00400014		add $a2, $s1, $0		a2=	
0x00400018		jal Proc1			
0x0040001C		addi $s2, $v0, $0		s2=	
0x00400020					
0x00400024	Proc1:	addi $sp, $sp, -12		sp=	
0x00400028		sw $ra, 8($sp)		Write the stored values on stack!	
0x0040002C		sw $s0, 4($sp)			
0x00400030		sw $s1, 0($sp)			
0x00400034		addi $v0, $0, 0			
0x00400038	Loop:	beq $a0, $0, Done			
0x0040003C		lw $s0, 0($a1)			
0x00400040		slt $s1, $s0, $a2			
0x00400044		beq $s1, $0, Next			
0x00400048		addi $v0, $v0, 1			
0x0040004C	Next:	addi $a0, $a0, -1			
0x00400050		addi $a1, $a1, 4			
0x00400054		j Loop			
0x00400058	Done:	lw $ra, 8($sp)		ra=	
0x0040005C		lw $s0, 4($sp)		s0=	
0x00400060		lw $s1, 0($sp)		s1=	
0x00400064		addi $sp, $sp, 12		sp=	
0x00400068		jr $ra			

Address	**Data**
0x100000000	13
0x100000004	10
0x100000008	21
0x10000000C	15
0x100000010	7
0x100000014	16
0x100000018	11
0x10000001C	6
0x100000020	30
0x100000024	28

Address	**Stack Data**
0x7FFFFFFC	XXXXXXXX
0x7FFFFFF8	
0x7FFFFFF4	
0x7FFFFFF0	

b) Briefly describe what Proc1 function does.