Alexandria University
Faculty of Engineering
Electrical Engineering Department
Midterm Exam, April 2016

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الكهربية
امتحان نصف الفصل الدراسي الثاني (إبريل ٢٠١٦)

Course Title and Code Number:
Computer Architectures (CS x35)
Fourth Year (Communications and Electronics)
Time Allowed: 90 minutes

اسم المقرر والرقم الكودي له:
معماريات الحاسب (CS x35)
السنة الدراسية الرابعة (اتصالات و الكترونيات)
الزمن: ٩٠ دقيقة

Answer the following questions in the dedicated space: **(25 marks)**

Question 1: **(9 marks)**

a) Implement the following high-level code segments using the `slt` instruction. Assume the integer variables `g` and `h` are in registers `$s0` and `$s1`, respectively.

i. `if (g > h)`
`g = g + h;`
`else`
`g = g - h;`	

ii. `if (g >= h)`
`g = g + 1;`
`else`
`h = h - 1;`	

iii. `if (g <= h)`
`g = 0;`
`else`
`h = 0;`	

b) Write a MIPS assembly program equivalent to the following pseudo-instructions. If necessary, you can use register `$t0` to memorize intermediary values. No other register can be used.

i. `add ($s0),$s1,($s2)        #mem[$s0]=$s1+mem[$s2]`

This MIPS instruction does not exist, because it uses an addressing mode not supported by RISC processors.

ii. `SWAP $t0, $t1              # $t0 <-> $t1`

This MIPS assembly does not exist; it is used to swap the contents of two registers, $t0 and $t1. You may not use any other registers

iii. `PUSH $s0`

This instruction is not a MIPS instruction either. It decrements the stack pointer (SP), then saves `$s0` at this address.

c) In MIPS assembly, write an assembly language version of the following C code segment:

```
for (i = 0; i < 98; i++) {
     C[i] = A[i + 1] - A[i] * B[i + 2]
     }
```

At the beginning of this code segment, the only values in registers are the base address of arrays A and B in registers `$a0` and `$a1`. Arrays A, B and C start at memory location *0xA000*, *0xB000* and *0xC000*, respectively. Try to reduce the total number of instructions and the number of expensive instructions such as multiplies.

Question 2: **(6 marks)**

Each number in the Fibonacci series is the sum of the previous two numbers. The following MIPS assembly code can be used to recursively find the Fibonacci function of an integer *n*.

```
fib:
addi $t0, $0, 2
slt $t0, $a0, $t0
beq $t0, $0, else
add $v0, $a0, $0
jr $ra
else:
sub $sp, $sp, 12
sw $ra, 0($sp)
sw $a0, 4($sp)
addi $a0, $a0, -1
```

```
#Continue
jal fib
sw $v0, 8($sp)
lw $a0, 4($sp)
addi $a0, $a0, -2
jal fib
lw $v1, 8($sp)
add $v0, $v0, $v1
lw $ra, 0($sp)
addi $sp, $sp, 12
jr $ra
```

Draw the status of the stack before calling `fib`, and during each function call for `$a0=4`. Indicate the registers and variables stored on the stack, mark the location of `$sp`, and clearly mark each stack frame.

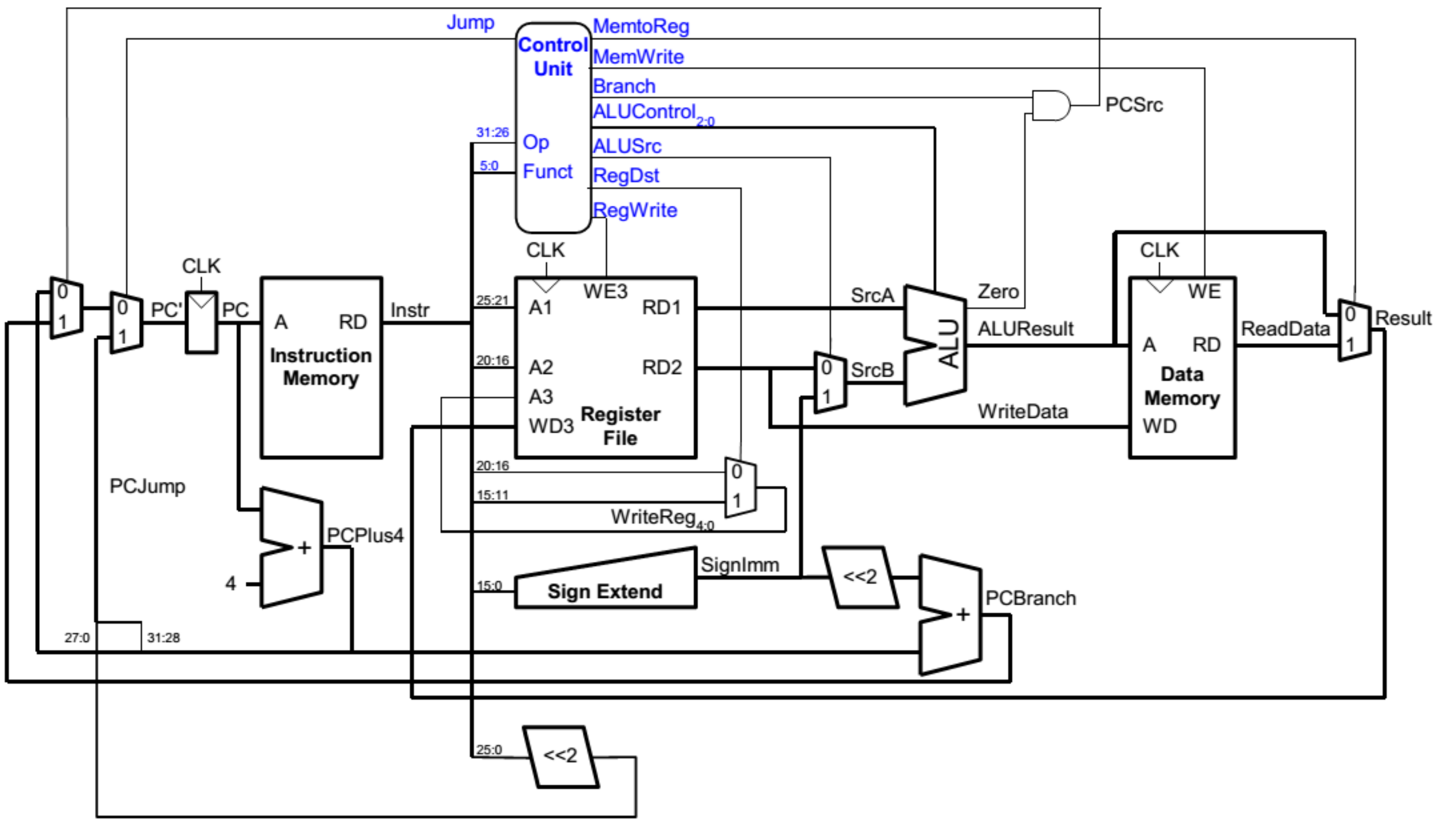
Question 3: **(6 marks)**

Modify the single-cycle MIPS processor to implement each one of the following instructions. Indicate the changes to the datapath on the following figure. Name any new control signals and complete the table to indicate changes required to the control signals. Also, indicate any changes needed to the ALU knowing that it only supports `add`, `sub`, `and`, `or`, and `slt` instructions.

a. `jal label           #jump and link $ra = PC + 4, PC = JTA`

b. `sllv $t0, $t1, $t2  #$t0 <= $t1 << $t2`

c. `jr rs               #jump register PC = [rs]`



Instruction	Op5:0	RegWrite	Reg Dst	Alu Src	Branch	Mem Write	MemtoReg	ALUOp 1:0	Jump		
R-type	000000	1	1	0	0	0	0	10	0		
lw	100011	1	0	1	0	0	1	00	0		
sw	101011	0	X	1	0	1	X	00	0		
beq	000100	0	X	0	1	0	X	01	0		
j	000100	0	X	X	X	0	X	XX	1		
jal	000011										
sll	100001										
jr	001000										

Indicate ALU changes if any:

Question 4: **(4 marks)**

a) An architecture engineer is contemplating building the single-cycle MIPS processor in a 65 nm CMOS manufacturing process. He has determined that the logic elements have the delays given in Table1. Help him compute the execution time for a program with 100 billion instructions.

Element	Parameter	Delay (ps)
register clk-to-Q	t_{pcq}	30
register setup	t_{setup}	20
multiplexer	t_{mux}	25
ALU	t_{ALU}	200
memory read	t_{mem}	250
register file read	$t_{RF\text{read}}$	150
register file setup	$t_{RF\text{setup}}$	20

b) The engineer is allowed to optimize only a single element to minimize its delay to the half. Which element should he choose and what will be the program execution time after this step.

Good Luck

Dr. Mohammed Morsy