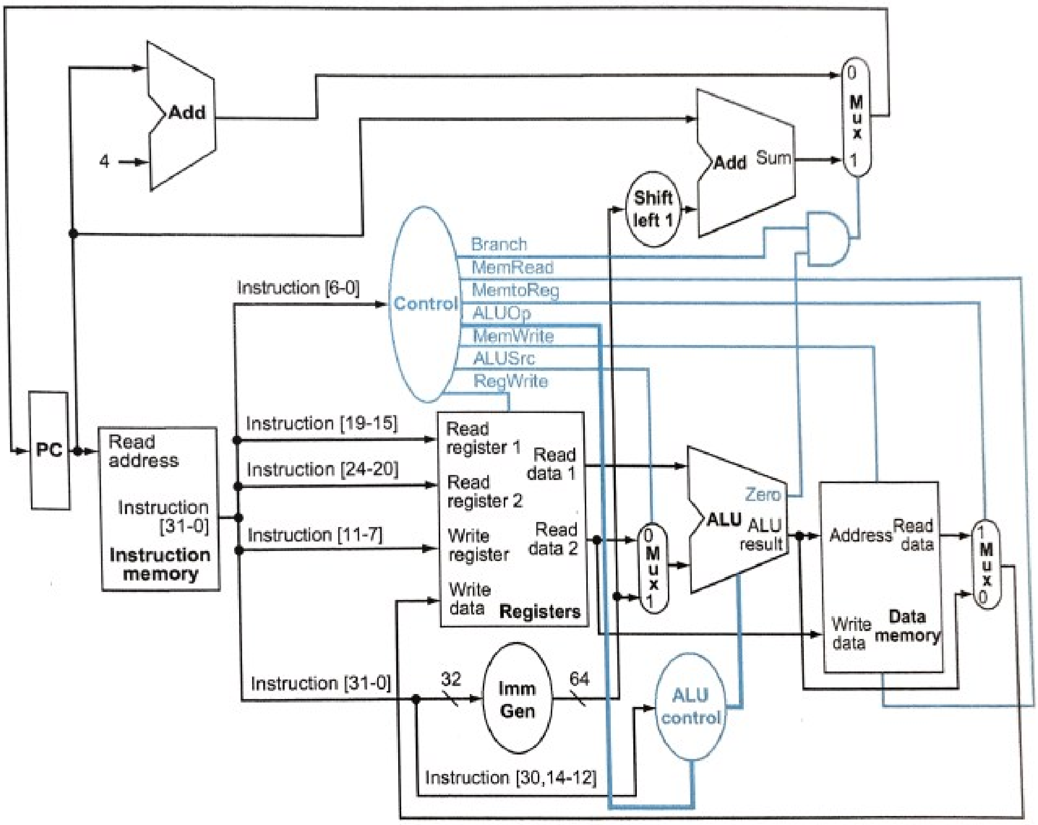
CSCI U 310 – Introduction to Computer Architecture

Homework-2 Key, Weights: 50 points

Due on Wednesday, February 6, 2019 at 5PM in Blackboard (a scan copy)

1. **(10 pts.)** The following processor diagram is the single cycle processor we covered in the class (Figure 4.24, textbook). This processor can handle MIPS R, I, and J type instructions.



The ALU control code is listed below.

0000:	and
0001:	or
0010:	add
0110:	subtract
0111:	slt
1100:	nor

With this the diagram above and the provided ALU code, complete the following table of control signal for MIPS/ RISC-V instruction `sw`.

MIPS Instruction	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALU Control Code (4 bits)
`sw`	1	X	0	0	1	0	0010

2. **(10 pts.)** Based on the same assumption of problem 1, complete the following table of control signal for MIPS/RISC -V instruction `addi`.

MIPS Instruction	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALU Control Code (4 bits)		
addi	1	0	1	0	0	0	0010		

3. **(10 pts.)** Based on the same assumption of problem 1, complete the following table of control signal for MIPS/RISC-V instruction `slt`.

MIPS Instruction	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALU Control Code (4 bits)		
slt	0	0	1	0	0	0	0111		

4. **(10 pts.)** Based on the same assumption of problem 1, complete the following table of control signal for MIPS/RISC-V instruction j.

MIPS Instruction	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALU Control Code (4 bits)		
j	X	X	0	0	0	X	XXXX		

5. **(10 pts.)** Someone wrote the following MIPS procedure, `test1`, whose argument is `$a0` and `$a1`. The procedure, `test1`, calls another procedure, `test 2` whose argument is `$a0`.

MIPS Instruction	ALUSrc	Memto-Reg	Reg Write	Mem Read	Mem Write	Branch	ALU Control Code (4 bits)		
slt	0	0	1	0	0	0	0111		

The following are the major body of the procedure, `test1`. Both of the return value of procedure `test1` and `test2` are stored in `$v0`.

```
test1:    addi $sp, $sp, K[1]
          (saving register section)
          move $s0, $a0
          move $s1, $a1
          addi $t0, $s0, 300
          move $a0, $t0
          jal test2
          add $v0, $v0, $s1
          (restoring register section)
          addi $sp, $sp, K
          jr $ra
```

Based on the procedure main body shown above, determine which registers need to store in the stack at the beginning of the procedure `test1` (this store function will be implemented in the 'saving register section') and the value of K.

[1]: Should be `addi $sp, $sp, -K`

[Solution]

The following registers need to be stored at the first:

`$ra` (by calling procedure test 2, `$ra` will be changed)
`$s0` (the main function might use this register)
`$s1` (the main function might use this register
Three registers need to be stored in memory, so K=3*4=12