Alexandria University
Faculty of Engineering
Computers and Communications Department
Mid-term Exam, March 2016

جامعة الإسكندرية
كلية الهندسة
قسم الهندسة الحاسبات والاتصالات
امتحان نصف الفصل الدراسي الثاني (مارس ٢٠١٦)

Course Title and Code Number:
Computer Architecture (CC 322)
Time Allowed: 1 hour

اسم المقرر والرقم الكودي له:
معماريات الحاسب (CC 322)
الزمن: ساعة واحدة

Fill in **the most appropriate choice** circle in the answer sheet. **Only one choice is allowed.** Choice **e** is none of the above**:** **(20 marks)**

<u>MIPS Instruction Set Architecture:</u>

1. "Simplicity favors regularity" is a design principle in the MIPS processor manifested by:
   a. Reduce the number of registers b. **Fix the number of operands**
   c. Reduce the number of instruction formats d. All of the above
2. "Make the common case fast" is a design principle in the MIPS processor manifested by:
   a. Reduce the number of registers b. Fix the number of operands
   c. **Reduce the number of instruction formats** d. All of the above
3. "Smaller is faster" is a design principle in the MIPS processor manifested by:
   a. **Reduce the number of registers** b. Fix the number of operands
   c. Reduce the number of instruction formats d. All of the above
4. The maximum number of the instructions that the MIPS processor can support using the three R-, I-, and J-Type instruction formats is:
   a. 63 instructions b. 64 instructions c. **127 instructions** d. 128 instructions
5. If the number of registers in a 32-bit MIPS processor is increased to 64 registers, the maximum number of instructions that this processor can support such that no immediate is represented in less than 16-bit width is:
   a. 23 b. 31 c. **47** d. 63
6. In the processor described in 5, the maximum number of R-type instructions is:
   a. 8 b. 16 c. **32** d. 64
7. The MIPS processor provides both `sll` and `sllv` instructions, which illustrates the following design principle(s):
   a. "Simplicity favors regularity" b. "Smaller is faster"
   c. **"Make the common case fast"** d. All of the above
8. To write a constant `a=0xFEDC8765` to the register `$S0`, the following instruction(s) can be used:
   a. `addi $s0, 0xFEDC8765`
   b. `ori $s0, $s0, 0x8765`
      `lui $s0, 0xFEDC`
   c. **`lui $s0, 0xFEDC`**
      `ori $s0, $s0, 0x8765`
   d. `li $s0, 0x1234AA77`
9. The following MIPS assembly program

```
# $s0 = g, $s1 = h
slt $t0, $s1, $s0
beq $t0, $0, else
add $s0, $s0, $s1
```

```
j done
else: sub $s0, $s0, $s1
done:
```

is the assembly description of the following C program:

a.
```
if (g < h)
  g = g + h;
else
  g = g - h;
```
b.
```
if (g <= h)
  g = g + h;
else
  g = g - h;
```
c.
```
if (g > h)
  g = g + h;
else
  g = g - h;
```
d.
```
if (g >= h)
  g = g + h;
else
  g = g - h;
```

10. The following assembly snippet describes a function called `proc1`:

```
proc1:
  add $t0, $a0, $a1
  add $t1, $a2, $a3
  sub $s0, $t0, $t1
  addi $v0, $s0, 0
  jr $ra
```

This assembly program is incomplete (the stack instructions are omitted). The size of the stack frame that should be saved by proc1 as a callee is:

a. 4 bytes b. 8 bytes c. 12 bytes d. 24 bytes

11. For the assembly program described in 10, if the following line is inserted before the `addi`:

```
jal proc2
```

The size of the stack frame that should be saved by proc1 as a callee is:

a. 4 bytes **b. 8 bytes** c. 12 bytes d. 24 bytes

12. For the assembly program described in 10, if the following line is inserted before the `addi`:

```
jal proc1
```

The size of the stack frame that should be saved by proc1 as a callee is:

a. 4 bytes **b. 8 bytes** c. 12 bytes d. 24 bytes

13. Calculate the immediate field for the `bne` instruction in the following program:

```
0xAC loop: add $t1, $a0, $s0
0xB0 lb $t1, 0($t1)
0xB4 add $t2, $a1, $s0
0xB8 sb $t1, 0($t2)
0xBC addi $s0, $s0, 1
0xC0 bne $t1, $0, loop
0xC4 lw $s0, 0($sp)
```

a. -5 b. 0xAC **c. -6** d. 0xB0

14. If the `0xC0 bne $t1, $0, loop` line in problem 13 is replaced with the following line `j loop`. Calculate the immediate field for the jump instruction.

a. -5 b. 0xAC c. -6 d. 0xB0

15. The following MIPS pseudo instruction `move $s2, $s1` can be implemented as:

a. `add $s2, $s1, $0` b. `sub $s2, $s1, $0`

c. `sllv $s2, $s1, $0` **d. All of the above**

16. The maximum offset of the `beq` instruction from the PC address in both directions is:

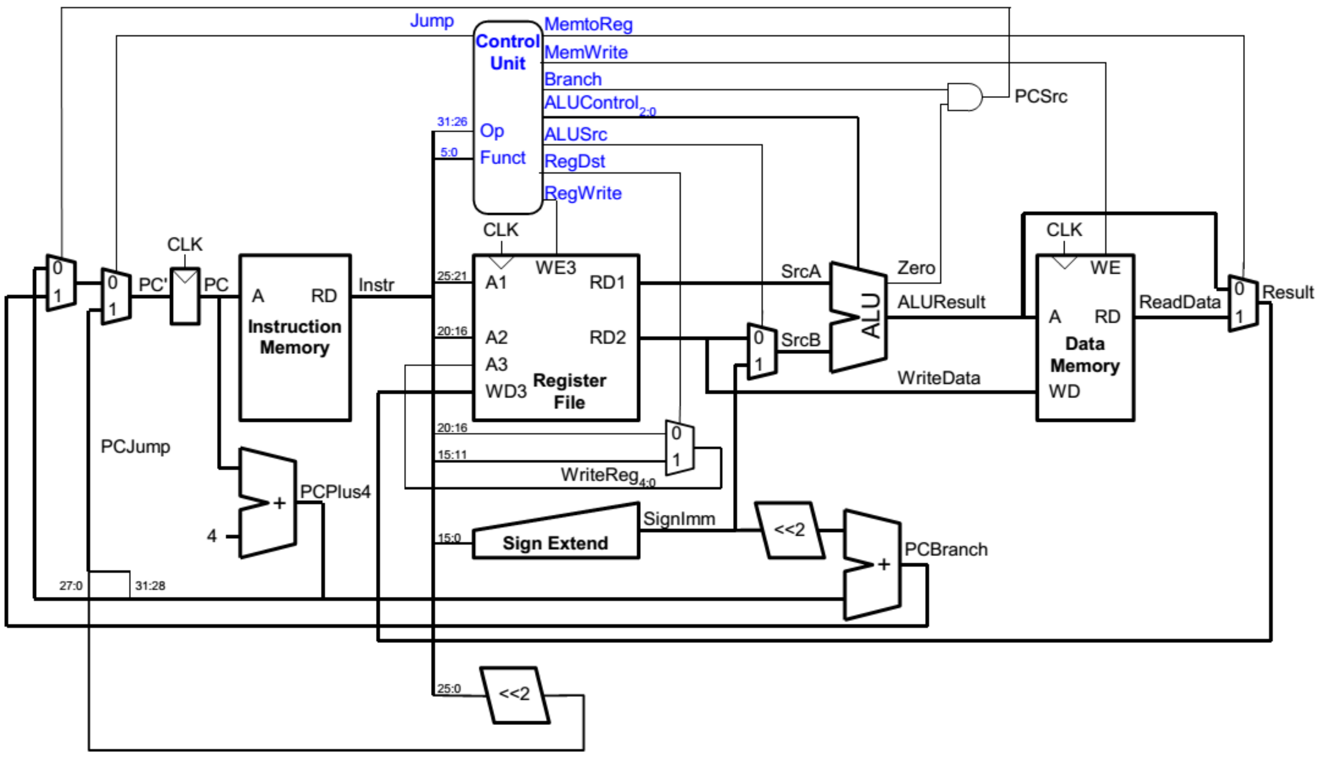
a. $\pm 2^{15}$ b. $\pm 2^{16}$ **c.** $\boldsymbol{\pm 2^{17}}$ d. $\pm 2^{18}$

17. The maximum offset (the best case) of the jump instruction from its current address in the forward direction is:
    a. 2^{25} b. 2^{26} c. 2^{27} **d. 2^{28}**
18. The minimum offset (worst case) of the jump instruction from its current address in the forward direction is:
    a. 2^{25} b. 2^{26} c. 2^{27} d. 2^{28}
19. How long will this program take to get to finish? (Assume each instruction takes 1 cycle)

```
start:
jal middle
finish:
middle:
jal last
jr $ra
last:
jr $ra
```

    a. 3 cycles b. 4 cycles c. 8 cycles d. **Forever**

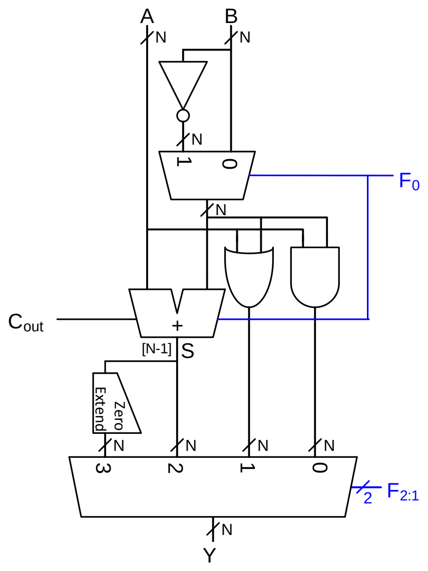
MIPS Single-Cycle Processor:



20. Which element in the processor is not a state element?
    a. Instruction memory b. Register file c. PC register d. Data memory
21. What control flow instructions does this datapath support?
    a. `beq, bne` b. `J, bne` c. **`J, beq`** d. `J, beq, bne`
22. What do you need to add to the datapath above to support the `jr` instruction?

a. A MUX and control logic to send the PC to the register file
b. **A MUX and control logic to send the register file data to the next PC**
c. Both (a) and (b)
a. Nothing needs to be changed

23. What needs to be changed in the processor shown above to support the `bne` instruction?
    a. Additional control signal of Branch_NE
    b. **Modify the PCSrc driving logic**
    c. Both (a) and (b)
    d. Nothing needs to be changed
24. Control signals need to be asserted by the `addi` instruction are:
    a. Regwrite=1, AluSrc=1, MemtoReg=1, others=0, and set the ALUOp to add.
    b. **Regwrite**=1**, AluSrc**=1**, others=0, and set the ALUOp to add.**
    c. Regwrite=1, RegDst=1, others=0, and set the ALUOp to add.
    d. Regwrite=1, RegDst=1, MemtoReg=1, others=0, and set the ALUOp to add.
25. Control signals need to be asserted by the `slt` instruction are:
    a. Regwrite=1, RegDst =1, others=0, and set the ALUOp to sub.
    b. Regwrite=1, AluSrc=1, others=0, and set the ALUOp to sub.
    c. **Regwrite=1, RegDst=1, others=0, and set the ALUOp to look at function.**
    d. Regwrite=1, AluSrc=1, others=0, and set the ALUOp to look at function.
26. The output of the ALU shown in Figure for F=111 is:
    a. `A&~B` b. **`slt`** c. `A|~B` d. `A-B`
27. The output of the ALU shown in Figure for F=101 is:
    a. `A&~B` b. `slt` c. `A|~B` d. **`A-B`**
28. The output of the ALU shown in Figure for F=110 is:
    a. `A&B` b. `slt` c. `A|B` d. `A+B`
29. For the shown ALU, the number of meaningful functions that can be used in the MIPS datapath is:
    a. `3` b. `4` c. **`7`** d. `8`



30. The exam level is:
    a. Easy
    b. Medium
    c. Difficult
    d. Very difficult