1. Which field determines the operation of an R-type instruction?

The FUNC field determines the actual operation (the opcode is 0 for all R-type instructions).

2. Suppose the program counter, PC, has the value 0x00001234. What is the value of PC after executing the following branch instruction?

PC = PC + 4 + sign-ext(4*imm)
= 0x1234 + 4 + 16 = 0x1248

3. Without making any assumptions about the contents of registers or memory, which of the following operations ***cannot*** be performed by a ***single*** MIPS instruction and why?

   (A) Memory[R[rs] + 0x1000] ←0
   (B) Memory[R[rs]] ←0
   (C) Memory[0x1000] ←0
   (D) Memory[R[rs] + R[rt]] ←0
   (E) R[rt] ←Memory[R[rs] + 0x1000]

The answer is (D) because there is no MIPS instruction that can add two registers and use the sum as a memory address. All other choices can be implemented using a swor lwinstruction.

4. Suppose you execute the following instruction sequence:

```
addi $t0, $0, -1
sll  $t0, $t0, 16
srl  $t1, $t0, 16
sra  $t2, $t0, 16
```

   What are the values of $t0, $t1and $t2after execution (in binary or hex)?

   $t0 = 0xFFFF0000

   $t1 = 0x0000FFFF

   $t2 = 0xFFFFFFFF

5. Assuming the standard MIPS procedure calling conventions, if we see an instruction of the form lw $t0, 4($fp), the program is most likely

   (A) accessing the return address
   (B) accessing one of its own local variables
   (C) accessing a local variable belonging to its caller
   (D) accessing its fifth argument
   (E) none of the above

The answer is (D) because the fifth argument (arg[4]) would typically be found one word above the location to which the frame pointer is pointing.

For the next two questions, consider the following assembly language procedure:

```
foo:
        addi $sp, $sp, -4 sw
                $ra, 0($sp) beq
                $a0, $0, L1 addi
        $a0, $a0, -1 add   $a1,
        $a1, $a1 jal         foo
        add     $a1, $v0, $0 L1:
add             $v0, $a1, $0
        lw   $ra,   0($sp)   addi
        $sp, $sp, 4 jr $ra
```

Suppose there is a procedure called mainwhich calls foo(4,3). [Assume that mainplaces 4 in $a0, and 3 in $a1before calling foo.]

6. What is the entire sequence of calls to foo,starting with foo(4,3)?
   foo(4,3) ,foo(3,6) , foo(2,12) ,foo(1,24) ,foo(0,48)
7. What is the final value returned to main?

The value returned to mainis 48.

8. The von Neumann model of a computer consists of three major components: the central processing unit (CPU), main memory, and input-output. In which of these components might we find the 32 registers of a MIPS processor?

A. CPU
B. Main memory
C. Input-output
D. CPU as well as main memory
E. Main memory as well as input-output

Correct answer is A. The registers are in the CPU, not in the memory.

9. Suppose we want to put the 32-bit value 0x456789AB into register **$4**. Which of the following is a good sequence of assembly instructions to do so?

A. **addi $4, $0, 0x4567**
sll $4, $4, 8
addi $4, $4, 0x89AB
B. **addiu $4, $0, 0x4567**
srl $4, $4, 8
addiu $4, $4, 0x89AB
C. **lui $4, 0x4567**
addi $4, $4, 0x89AB
D. **lui $4, 0x4567**
addiu $4, $4, 0x89AB
E. All of the above

Correct answer is D. C is incorrect because you need the addiu instruction (or, alternatively, ori) to ensure that the immediate is padded with zeros to its left, and not with its sign. A is incorrect because it uses addi for the 0x89AB part, and also because it shifts left by only 8 bits instead of 16. B is incorrect because it shifts right by 8 instead of left by 16.

10. Which of the following instructions, when encoded in binary, has a field that holds an immediate/constant value that is *NOT automatically multiplied by 4* during execution by the CPU:

A. **lw $4, 4($4)**
B. **sll $4, $4, 2**
C. **bne $4, $1, 0x0100**

D. **jal 0x0100**
E. The **lw**, **bne** and **jal** instructions above
F. The **lw** and **sll** instructions above
Correct answer is F. The lw instruction has an immediate value of 4, which is simply added to the value of register $4; there is no automatic multiplication by 4. The sll instruction has an immediate/constant value of 2, which is the number of bit positions by which the shifting takes places; there is no automatic multiplication of 2 by 4. The bne and jal instructions, on the other hand, contain an immediate that is automatically multiplied by 4 (to convert words to bytes) before being used to compute the jump/branch address. Hence, the answer is F. (Half credit if you choose only A or only B.)
11. Which of the following CANNOT be compiled as a single valid MIPS instruction:
A. **addu $s6, $s7, $t8**
B. **mul $2, $3, $4**
C. **lw $2, -3($4)**
D. **la $2, -3($4)**
E. None of the above
Correct answer is B. The mul instruction cannot be compiled as a single instruction. One would need a mult followed by an mflo, for example, to implement this instruction.
12. Which of the following addressing modes is available in MIPS?
A. Memory indirect
B. Displacement
C. Autoincrement
D. Indexed
E. Scaled
Correct answer is B. None of the others are available in MIPS.
13. The following five assembly statements reserve space for five different variables/arrays, named A, B, C, D and E. Circle the one that reserves a different amount of space than the other four:
A. **A: .byte 1, 2, 34, 45, 0, 0, 4, 8**
B. **B: .word 0, 0**
C. **C: .asciiz "Comp 411"**
D. **D: .asciiz "Quiz #2"**
E. **E: .space 8**
Correct answer is C. The string "Comp 411" needs 9 bytes (including its terminal NULL), while all others need 8 bytes.
14. Suppose *ProcedureA* calls *ProcedureB* with six arguments (*arg[0]—arg[5]*), and that registers **$sp** and **$fp** are properly managed by the assembly code. Within *ProcedureB*, how would the code read the argument *arg[5]* into register **$5**?
A. **lw $5, 4($fp)**
B. **lw $5, 8($fp)**
C. **lw $5, 5($sp)**
D. **lw $5, 20($sp)**
E. **ori $5, $a5, 0**
Correct answer is B. The spillover arguments (arg[4], arg[5], etc.) are stored in memory just above the location pointed to by the frame pointer. Therefore, arg[5] is at $fp+8.
15. Suppose the **main** part of your assembly code occupies the range of memory locations 0 to 63 (0x00000000 to 0x0000003F). Now, suppose within **main** you need to call a procedure that is located at the memory address 0xFFFFFF00 (really high address). Which of the following would be a good instruction to implement this procedure call?
A. **j**
B. **jal**
C. **jr**

D. **jalr**
E. **beq**
Correct answer is D. Since this is a procedure call, which requires the return address to be saved, the only candidates are jal and jalr. The jal instruction has a limitation that the 4 highest bits cannot be changed from their current value (see lecture slides). Thus, jalr must be used.