Q21.How many bits are required to identify a register available in MIPS architecture studied in class?

A. 2
B. 3
C. 4
D. 5

Answer：5

Q22. Assume $s1 contains memory address 0x0000 001C. Since memory addresses must be word-aligned, which one of the following will result in an error at run-time?

A. lw $t0, 6($s1)
B. sw $t0, 4($s1)
C. lw $t0, 4($s1)
D. lw $t0, 8($s1)
E. None of these

Answer: lw $t0, 6($s1)

Q23. Which of the following instructions computes the negation of the integer stored in $t0

A. sw $t0, -1($t0)
B. sub $t0, $t0, $zero
C. None of these
D. sub $t0, $zero, $t0
E. add $t0, $t0, $t0

Answer: sub $t0, $zero, $t0

Q24. After executing the following MIPS code, what is the value in $s1?

```
.data
ArrayA: .word 1 2 3 4 5

.text
  la    $s0, ArrayA
  addi  $s0, $s0, 4
  lw    $s1, 0($s0)
```

Note: ArrayA is the address of the first element of the array

A. 3
B. 2
C. None of these

D. 1

Answer: 2

Q25. Given a system which uses 12 bits to represent the memory addresses, and each byte in the memory has its own unique address, what is the maximum capacity of the memory?

A. 4,294,967,296 bits
B. 65,536 bits
C. None of these
D. 2,147,483,648 bits
E. 32,768 bits

Answer: 32,768 bits

Q26. Which of the following is the correct presentation of the following logic?
if(A>=B) goto Gteq

A. slt $t0, $s1, $s0
beq $t0, $0, Gteq
B. slt $t0 $s0, $s1
bne $t0, $0, Gteq
C. slt $t0, $s1, $s0
bne $t0, $0, Gteq
D. slt $t0, $s0, $s1
beq $t0, $0, Gteq

Answer: slt $t0, $s0, $s1
beq $t0, $0, Gteq

Q27. Which of the following is a pseudo-instruction?

A. sub $t0, $t1, $0
B. mov $to, $t1
C. ori $t2, $zero, 3
D. addi $t0, $t1, 10000
E. none of the above

Answer: mov $to, $t1
none of the above
Note: There are two typos in the instruction. The pseudo instruction is actually "move" not "mov". Therefore, the above two answers are both correct.

Q28. Which of the following code fragments will replace the value of X with that of Y, replace the value of Y with that of Z, and replace the value of Z with that of X?

i)

```
lw   $t0, ($s0)
lw   $t2, ($s2)
sw   $t0, ($s2)
lw   $t0, ($s1)
sw   $t2, ($s0)
sw   $t0, ($s1)
```

ii)

```
lw   $t0, ($s0)
lw   $t2, ($s2)
sw   $t0, ($s2)
lw   $t0, ($s1)
sw   $t0, ($s0)
sw   $t2, ($s1)
```

A. i and ii
B. i
C. ii
D. none of the above

Answer: ii

Q29. Consider the following MIPS code. Which pseudo-code represents what it does?
Assume $s0 = k, $s1 = g, $s2 = h, $s3 = f

```
    beq  $s0, $zero, L1
    li   $t0, 1
    beq  $s0, $t0, L2
    j    L3
L1: add  $s3, $s1, $s2
    j    End
L2: sub  $s3, $s1, $s2
    j    End
L3: add  $s3, $s1, $s1
    sub  $s3, $s3, $s2
End:
```

A.

```
if ( k != 0 ) {
    f = g - h ;
else if ( k != 1 ){
    f = g + h ;
else {
    f = g * g - h ;
}
```

B.

```
switch ( k ) {
  case 0:
    f = g + h ;
    break ;
  case 1:
    f = g - h ;
    break ;
  default:
    f = 2 * g – h ;
}
```

C. None of above

Answer:


```
switch ( k ) {
  case 0:
    f = g + h ;
    break ;
  case 1:
    f = g - h ;
    break ;
  default:
    f = 2 * g – h ;
}
```


Q30. Consider the MIPS procedure *Moo* below. Which pseudo-code represents what it does? Assume $ra contains the return address of the caller.

```
Moo: blt   $a0, $zero, Lbl
     move  $v0, $a0
     jr    $ra

Lbl: addiu $v0, $a0, 1
     jr    $ra
```

A. Moo(int n) { if (n >= 0) return (n+1) ; else return n ; }
B. Moo(int n) { if (n <= 0) return (n+1) ; else return n ; }
C. Moo(int n) { If (n < 0) return n ; else return n+1 ; }
D. Moo(int n) { If (n < 0) return (n+1) ; else return n ; }

Answer: Moo(int n) { If (n < 0) return (n+1) ; else return n ; }