CS2100 Computer Organization

2022/23 Semester 2

Midterm Assessment

SOLUTIONS

1. The following C program fragment inverts the bits in a uint32_t integer. The uint32_t datatype is similar to the int32_t datatype in Assignment 1, except that it is 32-bits long and unsigned. The example below shows what this operation is supposed to do. For brevity we will use a 16-bit number instead of a 32-bit number:

   Original: 0b1011 0010 0100 1100
   Return from invert: 0b0100 1101 1011 0011

   That is, all 0's are changed to 1's and all 1's are changed to 0's.

```
uint32_t invert(uint32_t x) {
    /* MISSING 1 */
}
```

   Which ONE of the following C statements should be inserted into the space marked /* MISSING 1*/? (Note: You are not allowed to enter and run the code on your computer. Doing so constitutes cheating)

   a. return -x;
   b. return !x;
   c. return x^0b0;
   d. +return ~x;
   e. None of the above options are correct.

2. The following C program fragment reverses the bits in a uint32_t integer. The example below shows how this operation works. Again, for brevity we will show a 16-bit integer instead of 32-bits:

Original: 0b0010 1101 0110 1010
Return from reverse: 0b0101 0110 1011 0100

```c
uint32_t reverse(uint32_t x) {
    uint32_t result = 0;

    for(int i=0; i<32; i++) {
        /* MISSING 2 */
    }

    return result;
}
```

Which ONE of the following C statements should be inserted into the space marked /* MISSING 2*/? (Note: You are not allowed to enter and run the code on your computer. Doing so constitutes cheating)

a. result |= (x & (0b1 << i)) << (31 - i);
b. +result |= (x & (0b1 << i)) << (31 - 2*i);
c. result |= (x & (0b1 >> i)) >> (31 - 2*i);
d. result &= (x & (0b1 << i)) << (31 - i);
e. None of the above options a. to d. are correct.

3. We now call both the earlier invert and reverse functions in a function called “combine”, to produce the behavior shown below (example this time is in 32-bits):

```
Original: 00000000000000000010110101101010
Combining the two functions: 10101001010010111111111111111111
```

The combine function is shown here:

```
uint32_t combine(uint32_t x) {
    /* MISSING 3 */
}
```

4. Which ONE of the following C statements can be inserted into the space marked /* MISSING 3 */? (Note: You are not allowed to enter and run the code on your computer. Doing so constitutes cheating)

   a. return !invert(reverse(x));
   b. return !reverse(invert(x));
   c. return ~invert(reverse(x));
   d. return ~reverse(invert(x));
   e. +None of the options a. to d. are correct.

5. Consider a 16-bit fixed-point number system, in sign and magnitude representation, with 8 bits in the integer portion (including sign bit) and 8 bits in the fraction portion. Fill in the following blanks:

   Most positive number that can be represented: **127.99609**
   Most negative number that can be represented: **-127.99609**
   Smallest positive number that can be represented: **0.0039063**

   Consider again the same 16-bit fixed point number system, in sign-and-magnitude representation with 8 bits in the integer portion (including sign bit) and 8 bits in the fraction portion. Find the ABSOLUTE ERROR for representing the numbers 3.625 and 9.151. When representing both numbers, consider possible rounding of the least significant bit.

   3.625 = 00000011.10100000_2
   Absolute error = 0

   9.151 = 00001001.001001101 = 00001001.00100111_2 = 9.1523438
   Absolute error in representing 3.625: **0**
   Absolute error in representing 9.151: **0.001344**

6. Given the same 16-bit fixed point number system in sign-and-magnitude representation with 8 bits for the integer portion (including sign bit) and 8 bits for the fraction portion, what is 0x95B0 in base-10?

   1001 0101 1011 0000
   = 0x95B0
   = -21.6875

7. What is $1213_5 + 312_5$ in base 5?

   $1213_5 + 312_5$ = **2030** in base 5.

8. What is $114_7 – 36_7$ in base-7? (Hint: Think about how you'd subtract two base-10 numbers, and apply the same idea here)

   $114_7 – 36_7$ = **45** in base 7.

9. $312_x = 431_5$. What is the mystery base x?

   x = **6**

10. What is -19.1015625 in IEEE-754 format? Write your answer in hexadecimal.

    19.1015625 = 10011.0001101 x 2^0
    1.00110001101 x 2^4
    Mantissa = 00110001101000000000000
    Exponent = 4 + 127 = 131 = 10000011
    Sign = 1
    Total = 11000001100110001101000000000000
    = 1100 0001 1001 1000 1101 0000 0000 0000
    = 0xC198D000

The MIPS code fragment below works on an integer array *A* whose base address is stored in **$s0** and size (number of elements) of the array in **$s1**. The first instruction **addi $t1, $s0, 0** is at address 0x00400034.

```
      addi $t1, $s0, 0       # Inst1 @ 0x00400034
      lw   $t2, 0($t1)       # Inst2
      addi $t0, $zero, 1     # Inst3
Loop: beq  $t0, $s1, Exit    # Inst4
      addi $t1, $t1, 4       # Inst5
      lw   $t3, 0($t1)       # Inst6
      add  $t3, $t3, $t2     # Inst7
      sw   $t3, 0($t1)       # Inst8
      addi $t2, $t3, 0       # Inst9
      addi $t0, $t0, 1       # Inst10
      j    Loop              # Inst11
Exit:
```

11. Below is an equivalent code in C. Let the array name be *A* and the size (number of elements) of *A* be *size*. `int i;`

```
for (i = 1; i < size; i++) {
    A[i] = A[i-1] + A[i];
}
```

12.

a. Assuming that *size* is 8 and array *A* contains the following valu es: 3, 5, -2, 4, 0, 7, -1, 6 (*A*[0]=3 and *A*[7]=6). What is the final value of **$t3**?

Answer: **22** (Array *A* after execution: 3, 8, 6, 10, 10, 17, 16, 22.)

b. For the instruction **beq $t0, $s1, Exit**, write out its hexadecimal representation.

Answer: **1 1 1 1 0 0 0 7**

opcode = 4; $t0 = $8; $s1 = 17; Exit = 7; beq $t0, $s1, 7 → 4, 8, 17, 7

4, 8, 17, 7 → 0001/00, 01/000, 1/0001/, 0000/0, 000/00, 00/0111

→ **1 1 1 1 0 0 0 7**

c. For the instruction **sw $t3, 0($t1)**, write out its hexadecimal representation.

Answer: **A D 2 B 0 0 0 0**

opcode = 2b; $t3 = $11; $t1 = 9; sw $t3, 0($t1) → 2b, 9, 11, 0

2b, 9, 11, 0 → 1010/11, 01/001, 0/1011/, 0000/0, 000/00, 00/0000

→ **A D 2 B 0 0 0 0**

d. For the instruction **j Loop**, write out its hexadecimal representation.

Answer: **0 8 1 0 0 0 1 0**

opcode = 2; address of Inst4: 0x00400040

0000/10 ~~0000~~ 00/00 01/00 00/00 00/00 00/00 01/00 00~~00~~

→ **0 8 1 0 0 0 1 0**

Expected wrong answer: address of Inst4: 0x00400037

0000/10 ~~0000~~ 00/00 01/00 00/00 00/00 00/00 00/11 01~~11~~ → 0 8 1 0 0 0 0 D

Expected wrong answer:

0000/10 00/00 00/00 01/00 00/00 00/00 00/00 01/~~00 0000~~ → 0 8 0 1 0 0 0 1

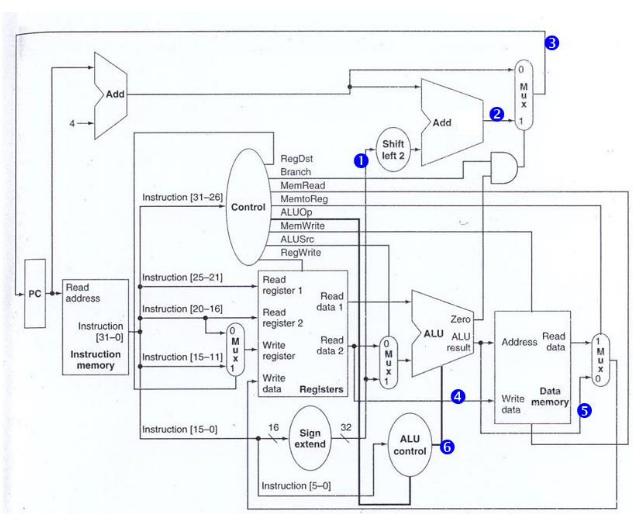
Expected wrong answer: use relative displace (as in branch) -8

-8 = 11 1111 1111 1111 1111 1111 1000 (-8 in 2's complement)

000010 11 1111 1111 1111 1111 1111 1000 → 0 B F F F F F 8

Consider the MIPS datapath with the values of the registers shown in the table below, where all values are in hexadecimal. Suppose the instruction **sw $a3, -20($s2)** at address 100_{10} is currently being executed.

```
R0  (r0) = 0x00000000 | R1  (at) = 0x00002000
R2  (v0) = 0x00000321 | R3  (v1) = 0x0000000A
R4  (a0) = 0x00000005 | R5  (a1) = 0x7FFFF000
R6  (a2) = 0x7FFFF004 | R7  (a3) = 0x700003AC
R8  (t0) = 0x00000001 | R9  (t1) = 0x00000c00
R10 (t2) = 0x0000C000 | R11 (t3) = 0xfffffff0
R12 (t4) = 0xF0000000 | R13 (t5) = 0x00000fff
R14 (t6) = 0x00006200 | R15 (t7) = 0x00000e00
R16 (s0) = 0x00300000 | R17 (s1) = 0x00000c00
R18 (s2) = 0xCC040200 | R19 (s3) = 0x000E5003
R20 (s4) = 0xCC030200 | R21 (s5) = 0x10000000
R22 (s6) = 0x00055000 | R23 (s7) = 0xF0000000
R24 (t8) = 0x00000005 | R25 (t9) = 0x0000D624
R26 (k0) = 0x00000000 | R27 (k1) = 0x00000000
R28 (gp) = 0x10008000 | R29 (sp) = 0x7FFFEff4
R30 (s8) = 0x1000000F | R31 (ra) = 0x00400018
```



13. What is the value at ❶ (output from sign-extend)? Your answer must be in hexadecimal.

Answer: **0xFFFFFFEC**

20_{10} = 0000 0000 0000 0000 0000 0000 0001 0100_{2s}

-20_{10} = 1111 1111 1111 1111 1111 1111 1110 1100_{2s}

A. 0x00000014

B. 0x0000FFEC

C. 0xFFFFFFEB

D. **+0xFFFFFFEC**

E. None of the above.

14. What is the value at ❷? Your answer must be in hexadecimal. (Taker

Answer: **0x00000018**

Note: Answer without leading zeroes is acceptable.

Immediate value = -20_{10} = 1111 1111 1111 1111 1111 1111 1110 1100_{2s} (from part a)

Immed << 2 = 1111 1111 1111 1111 1111 1111 1011 0000

PC+4 = 100_{10} + 4 = 68_{16} = 0x0000 0000 0000 0000 0000 0000 0110 1000

Adding, we have 0x0000 0000 0000 0000 0000 0000 0001 1000

A. **+0x00000018**

B. 0x00000050

C. 0x00000054

D. 0x80000054

E. None of the above.

15. What is the value at ❸? Your answer must be in hexadecimal. (Taken)

Answer: **0x00000068**

Note: Answer without leading zeroes is acceptable.

PC+4 = $100_{10} + 4 = 68_{16}$ = 0x0000 0000 0000 0000 0000 0000 0110 1000

A. 0x00000064

B. **+0x00000068**

C. 0x00000100

D. 0x00000104

E. None of the above.

16. What is the value at ❻ (ALU control output)? Your answer must be in binary. (Taken)

Answer: **0b0010**

sw requires ALU to perform an add operation. The 4-bit ALUcontrol for add is 0010.

A. 0b0000

B. 0b0001

C. **+0b0010**

D. 0b0110

E. None of the above.