**CS2115 2023/2024 Midterm Examination**

**Name: SID:**

**Total points: 100**

**Time: 2 hours**

**Section A (20 points).** *2 points for each question. Only one correct answer for each question.*

1. The octal representation of binary number 1001 0010 is \_\_\_\_\_.

A. (221)8

B. (222)8

C. (223)8

D. (224)8

2. When we perform **(-2)10 + (-5)10**, the answer in 4-bit 2’s complement is \_\_\_\_\_.

A. 1001

B. 1110

C. 1010

D. 0010

3. The 8-bit 2’s complement number 10010011 is equivalent to the decimal number \_\_\_\_\_\_.

A. -109

B. 109

C. -108

D. 108

4. The binary number of 9.1875 is \_\_\_\_\_.

A. 1001.1001

B. 1001.0101

C. 1001.0011

D. 1001.0110

5.Given the following truth table, where A and B are the inputs, C is the output. Then the corresponding logic gate is \_\_\_\_\_.

|  |  |  |
| --- | --- | --- |
| A | B | C |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

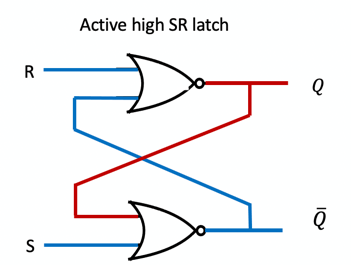
A. NAND

B. OR

C. NOT

D. NOR

6. For an ***Activate High SR Latch***, R and S are inputs, *Q* and are outputs. When R and S are set to\_\_\_\_ respectively, the output states of Q and will depend on the past state.



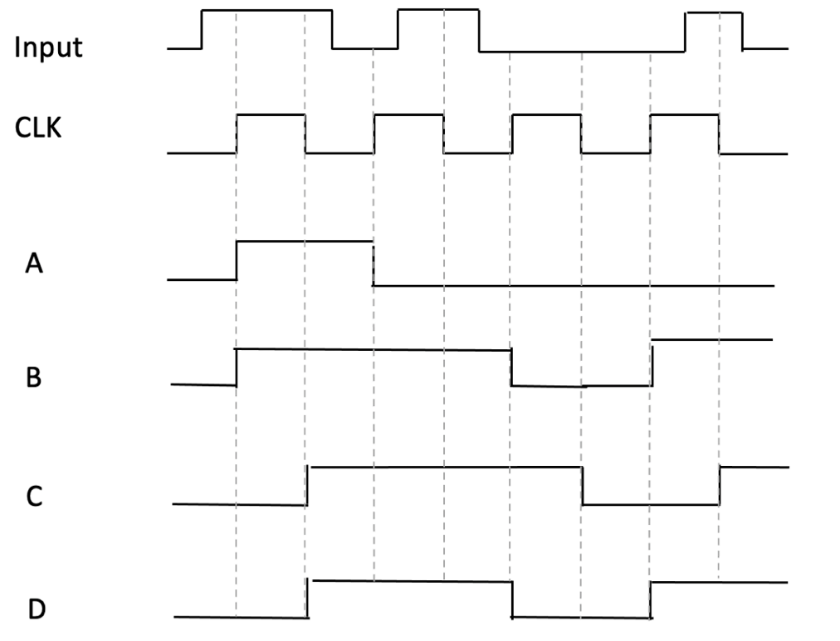
A. 0,1

B. 1,0

C. 0,0

D. 1,1

7. Suppose the timing diagrams of the input I and clock (CLK) of a D flip-flop triggered by ***rising edges*** are as follows, then the timing diagram of its output Q is \_\_\_\_\_\_\_.



8. In MIPS, operands are stored in \_\_\_\_\_.

A. computation unit registers

B. memory

C. registers

D. input device

9. In the aligned addresses mode, which of the following could be the starting address of access to a word in the memory? \_\_\_\_\_.

A. 0

B. 7

C. 2

D. All of above.

10. The following MIPS assembly program is to compute 5*x*2 + 2*x* + 3, where *x*=10. Please select the correct to be filled in the blank to complete the program. \_\_\_\_\_

.data

x: .word 10

.text

main:

lw $t0, x

mult $t0, $t0

mflo $t1

li $t3, 5

mult $t1, $t3

mflo $t2

li $t5, 2

mult $t0, $t5

mflo $t4

li $t6, 3

add $s1, $t2, $t4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ***(Fill the blank)***

li $v0, 10

syscall

A. add $t0, $t1, $t2

B. add $s1, $t0, $t4

C. add $t2, $s1, $t6

D. add $s1, $s1, $t2

**Section B (20 points).** *4 points for each question.*

1. What is the BCD (Binary Coded Decimal) representation of decimal number 893? Please answer the question and write the procedure of your reasoning.
2. What is the hexadecimal representation of decimal number 165.75? Please answer the question and write the procedure of your calculation.
3. Please calculate the result of (1A)16+(36)16 and convert the result to octal representation. Please write the procedure of your calculation.
4. What is the 8-bit 2’s complement representation of the result of (17)8×(-3)10 ? Please answer the question and write the procedure of your calculation.
5. Suppose a 32-bit floating-point number has one *sign* bit, 8 *biased exponent* bits and 23 *significand* bits. What is the floating-point representation of decimal number **-133.875**? Please answer the question and write the procedure of your calculation. (Hint: biased exponent E = real exponent value E’ + bias, where bias = 2K-1-1, K is the number of biased exponent bits.)

**Section C (25 points).**

**Q1.(3 points)** Given the following truth table, please draw the Karnaugh Maps and derive the corresponding logic expressions for the output D.

|  |  |  |  |
| --- | --- | --- | --- |
| **Inputs** | | | **Output** |
| **A** | **B** | **C** | **D** |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 |

**Q2. (4 points)** Given the following truth table, please draw the Karnaugh Maps and derive the corresponding logic expressions for the outputs F.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | | | | **Output** |
| **A** | **B** | **C** | **D** | **F** |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

**Q3.(6 points)** Please complete the the truth table of the following circuit, where A and B are inputs; C, D and E are outputs.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Truth table | | | | | | |  |
| **Input** | | **Output** | | |
| **A** | **B** | **C** | **D** | **E** |
| **0** | **0** |  |  |  |
| **0** | **1** |  |  |  |
| **1** | **0** |  |  |  |
| **1** | **1** |  |  |  |

**Q4.(6 points)** Suppose the timing diagram of the input D and clock CLK of a D flip-flop triggered by ***falling edges*** is as follows, please draw the timing diagram of its output Q.

**A white rectangular object with black lines

Description automatically generated**

**Q**

**Q5.(6 points)** The following figure shows the circuits of a 4-bit carry look ahead adder (CLA). The inputs are two 4-bit numbers, A3A2A1A0 and B3B2B1B0, as well as a carry-in bit C0. If A3A2A1A0=0110, B3B2B1B0=1001, and C0=1, then use the CLA carry-out equation to compute the values of following outputs: (i) carry-out C3, (ii) sum S3. **Please write down the detailed steps leading to your answer, i.e., the equations to calculate all outputs. If you only give the result but no detailed steps are provided, you will receive 0 points. (Note: Gi=AiBi, Pi=Ai**⊕**Bi, Ci=Gi-1+Pi-1Ci-1, Si=Ai**⊕**Bi**⊕**Ci)**

**A diagram of a logic

Description automatically generated**

**Section D (35 points).**

**The following table lists the MIPS instructions that may be useful (you can also use other MIPS instructions not included in this Table):**

|  |  |  |
| --- | --- | --- |
| **Name** | **Format** | **Functionality** |
| li | li $r1, x | load immediate number x to one register ($r1) |
| la | la $s1, Label | load the address of Label to register $s1 |
| lw | lw $s1, 10($t1) | load a word from memory with address (10+$t1) to register $s1 |
| sw | sw $s1, x | store data from $s1 to x |
| add | add $r1, $r2, $r3 | add two register numbers ($r2 + $r3) and store the result in a register ($r1) |
| addi | addi $t2, $t0, 3 | add $t0 with 3 and store the result in $t2 |
| sub | sub $r1, $r2, $r3 | sub two register numbers ($r2 - $r3) and store the result in one register ($r1) |
| mult | mult $r1, $r2 | multiply two register numbers ($r1 and $r2) and store the result in the [**hi**, **lo**] register pair (high bits stored in **hi** and low bits stored in **lo**) |
| move | move $r1, $r2 | move the value of one register ($r2) to another register ($r1) |
| mflo | mflo $r1 | move the value from lo register to the $r1 register |
| mfhi | mfhi $r1 | move the value from hi register to the $r1 register |
| beq | beq $r1, $r2, Label | if the first register number ($r1) equals the second register number ($r2), then jump to Label |
| blez | blez $r1, Label | if the register number ($r1) is less than or equal to zero, then jump to Label |
| bne | bne $r1, $r2, Label | if the register number ($r1) is not equal to the register number ($r2), then jump to Label |
| bnez | bnez $r1, Label | if the register number ($r1) is not equal to zero, then jump to Label |
| bgtz | bgtz $r1, Label | if the register number ($r1) is greater than zero, then jump to Label |
| bge | bge $t0, $t1, Label | If $t0 is greater or equal to $t1, then jump to Label |
| srl | srl $r1, $r2, n | logical right shift the value of one register ($r2) by n bits and store the result in another register ($r1), i.e., unsigned division by 2n |
| sll | sll $r1, $r2, n | logical left shift the value of one register ($r2) by n bits and store the result in another register ($r1), i.e., multiplication by 2n |
| slt | slt $r1, $r2, $r3 | set $r1=1 if $r2<$r3 |
| jal | jal ProcedureAddr | jump to the starting address ProcedureAddr of the procedure; store the return address into $ra. The parameters should be put in $a0-$a3 |
| j | j Label | directly jump to the given label without needing to satisfy any condition |
| jr | jr $ra | the jump register instruction, jump back to the caller when the procedure finishes. The return values should be stored in $v0-$v1 |

**Q1**. **(2 points)** Explain what is the main functionality of J-Type instructions in the MIPS.

**Q2**. **(4 points)** Explain the main difference between R-type and I-type instructions in MIPS (in terms of the functionalities of instructions in each type).

**Q3**. **(4 points)** In MIPS, the five steps required to execute an instruction are:

1.

2. Figure out what operation to do​

3.

4. Do the computation​

5. Figure out the next instruction

**Q4**. **(9 points)** The formate of R-type instructions in MIPS is

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Opcode  (6 bits) | 1st source  (5 bits) | 2nd source  (5 bits) | Destination  (5 bits) | Shamt code  (5 bits) | Funct code  (6 bits) |

Consider the following MIPS assembly instruction:

**add $t0, $t1, $t2**

where $t1 is the first source register, $t2 is second source register, and $t0 is destination register. In MIPS assembly, registers are identified by their numbers, and each register number corresponds to a specific register. The operation code, shamt code and function code of the **add** are,

|  |  |
| --- | --- |
| Opcode | 000000 |
| Shamt code | 00000 |
| Funct code | 100000 |

and the register numbers of $t0, $t1, $t2 are listed in the following table:

|  |  |
| --- | --- |
| Register | Register number |
| $t0 | 8 |
| $t1 | 9 |
| $t2 | 10 |

Please write the **hexadecimal** representation of this MIPS instruction (**add $t0, $t1, $t2**). Please briefly explain the procedure leading to your answer.

**Q5. (6 points)** Complete the following MIPS assembly program to calculate the sum of the first ***n*** positive integers and print the result, which execute the following main steps:

1. Accept an integer input ***n*** from the user (assume ***n*** is positive and not very large, e.g., <100).

2. Calculate the sum of the first ***n*** positive integers.

3. Print the result.

.data

n: .word 0 # Initialize n to store user input

prompt: .asciiz "Enter a positive integer (n): "

.text

main:

# Print the prompt

li $v0, 4 # Load syscall code 4 (print string)

la $a0, prompt

syscall

li $v0, 5

syscall

move $t0, $v0 # Store the user input in $t0 (n)

# Calculate the sum

li $t1, 0 # Initialize $t1 to 0 (sum)

li $t2, 1 # Initialize $t2 to 1 (counter)

calculate\_sum:

# Check if the counter (t2) is greater than n

bgt $t2, $t0, print\_result

# Add counter to sum

(***please fill this blank***)

# Increment the counter

(***please fill this blank***)

# Repeat the loop

j calculate\_sum

print\_result:

# Print the sum result

li $v0, 1 # Load syscall code 1 (print integer)

move $a0, $t1 # Load the result into $a0

syscall

**Q6. (10 points)** Write a complete MIPS assembly program to find and print the maximum element in an array of integers.

1. Initialize an array of integers named ***myArray*** with the following values: [15, 7, 23, 4, 12, 9, 18, 31, 2, 19].

2. Determine the maximum element in the array and store it in a variable named maxValue.

3. Print the maxValue to the console.

.data

myArray: .word 15, 7, 23, 4, 12, 9, 18, 31, 2, 19

arraySize: .word 10

maxValue: .word 0

prompt: .asciiz "The maximum value is: "

.text

main:

(***Write your answer here***)

# Print the maximum value

li $v0, 4 # Load syscall code 4 (print string)

la $a0, prompt # Load the address of the prompt

syscall

li $v0, 1 # Load syscall code 1 (print integer)

move $a0, $t1 # Load the maxValue into $a0

syscall

# Exit the program

li $v0, 10 # Load syscall code 10 (exit)

syscall

**END**