CDA 3103 Computer Organization Final Exam Practice

**Name: USF ID:**

**Exam Rules**

* Exam time: 120 mins
* Make sure that your writing is legible; otherwise your grades may be adversely affected.
* Close book, notes and HW.
* All electronics must be turned off.
* Show all work to get **partial credits** except yes/no problems.

**Important Reminder on Academic Honesty**

Using unauthorized information or notes on an exam, peeking at others work, or altering graded exams to claim more credit are severe violations of academic honesty. Detected cases will receive a failing grade in the course.

**Problem 1 [20 points] Multiple Choice Questions**

Please refer to quizzes, midterm 1 and 2 multiple choice questions.

**Problem 2 [20 points] Fill in Blanks**

1. Review questions for Module 1 and 2:
   1. Computer building components
   2. Left and right shift operations
   3. Sign extension operations
   4. Data range for n-bit unsigned numbers, sign-magnitude, 1’s complement and 2’s complement
   5. Binary number representations in sign-magnitude, 1’s complement and 2’s complement
   6. Moore’s law and Rock’s law
2. Suppose a computer has bytes of byte-addressable main memory and a cache size of bytes, and each cache block contains 32 bytes.
3. How many blocks of main memory are there? ( )
4. How many blocks of cache memory are there? ( )
5. If this cache is direct-mapped, what is the format of a memory address as seen by the cache; that is, the size of the tag field , the size of block field , and the size of the offset field .
6. If this cache is fully associative, what is the format of a memory address as seen by the cache; that is, the size of the tag field , and the size of the offset field .
7. If this cache is 4-way set associative, what is the format of a memory address as seen by the cache: the size of the tag field , the size of set field , and the size of the offset field .
8. Suppose you have a byte-addressable virtual address memory system with 8 virtual pages of 64 bytes each, and 4-page frames. Assuming the following page table, answer the questions below:
   1. How many bits are in a virtual address?
   2. How many bits are in a physical address?
   3. How many bits are virtual page number?
   4. How many bits are page frame number?
   5. How many bits are page offset?

**Problem 3 [15 points] Short Answer Questions. Show procedures to earn full credits.**

1. The numbers shown below are in 2’s complement binary representation. Determine whether the indicated arithmetic operations produce an overflow.
2. 0011 01102 + 1110 00112
3. 1001 10002 – 0010 00102
4. 0011 01102 – 1110 00112

|  |  |  |
| --- | --- | --- |
| a) | b) | c) |

1. Suppose the cache access time is 10ns, main memory access time is 200ns, the hard drive access time is 10ms, the TLB hit rate is 98%, the cache hit rate is 95% and the page fault rate is 0.001%. Assuming non-overlapped access, what is the average access time for the processor to access an item?

**Problem 4 [10 points] Combinational Circuits**

1. Write down **Canonical sum of product and Canonical product of sum Boolean functions** for the blow circuit.

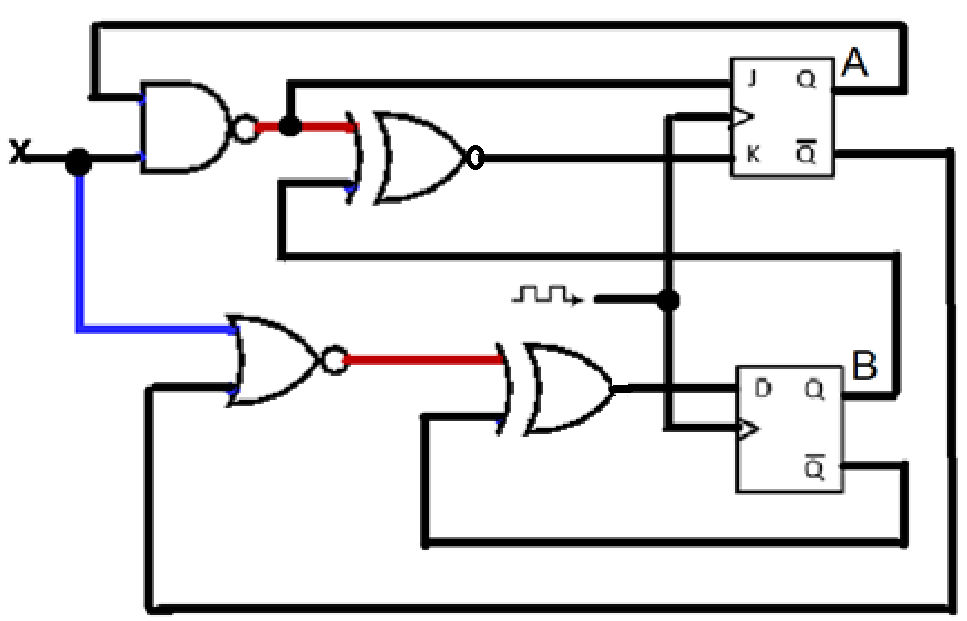
Diagram

Description automatically generated

1. Implement the circuit in problem 4a) with a 4 X 16 decoder and some necessary logic gates.
2. Implement the circuit in problem 4a) with a 16 to 1 multiplexer and some necessary logic gates.

**Problem 5 [10 points] Sequential Circuit Analysis**

Given the following sequential circuit,



Complete the truth table for the following sequential circuit:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Current States | | X | J | K | D | Next States | |
|  |  |  |  |
| 0 | 0 | 0 |  |  |  |  |  |
| 0 | 0 | 1 |  |  |  |  |  |
| 0 | 1 | 0 |  |  |  |  |  |
| 0 | 1 | 1 |  |  |  |  |  |
| 1 | 0 | 0 |  |  |  |  |  |
| 1 | 0 | 1 |  |  |  |  |  |
| 1 | 1 | 0 |  |  |  |  |  |
| 1 | 1 | 1 |  |  |  |  |  |

**Problem 7 [10 points] Convert the following high-level code into RISC-V assembly language.**

1. Write RISC-V instructions for placing the following immediate constants in register s0. Use a minimum number of instructions. **Pseudo-instructions are not allowed.**
2. 0xABCD9BCD
3. 0xABCD9235
4. Assume that the signed integer variables option and **ampt** are in registers **t0** and **t1**, respectively.

**if ( option == 1 )**

**amt = 200;**

**else if ( option == 2 )**

**amt = 500;**

**else if ( option == 3 )**

**amt = 1000;**

**else**

**amt = 0;**

1. Assume a string **str** and each character is stored as one byte. Also assume that the array base address is stored in register **a0**, the index variable **i** is stored in register **t0,** and the count variable **count** is stored in the register **t1**.

**if ((str[i] >= ‘a’ && str[i] <= ‘z’) || (str[i] >= ‘A’ && str[i] <= ‘Z’))**

**count = count + 1;**

1. Assume that the integer variable **i** is stored in registers **t0**. To display the value stored in register **t0** on the console, you need service number 1 of system calls.

**addi a7, zero, 1 # service 1 is print integer**

**add a0, t0, zero # load desired value into argument register a0**

**ecall**

**while ( i != 1)**

**{**

**printf(“%d ”, i);**

**if (i % 2 == 0) {**

**i = i/2;**

**}**

**else {**

**i = 3 \* i + 1;**

**}**

**}**

**Problem 8 [15 points] RISC-V Assembly Programming**

Rewrite the following high-level code to a RISC-V assembly function. The function **int getMedian(int ar1[], int ar2[], int n)** find the median of two sorted arrays of same size. Assume **a0, a1,** and **a2** hold base address of **ar1** array, **ar2** array and the array size, respectively. Two arrays are already sorted.

**int getMedian(int ar1[], int ar2[], int n)**

**{**

**int i = 0;  /\* index for array ar1[] \*/**

**int j = 0; /\* index for array ar2[] \*/**

**int count;**

**int m1 = -1, m2 = -1;**

**/\* Since there are 2n elements, median will be average of elements at**

**index n-1 and n in the array obtained after merging ar1 and ar2 \*/**

**for (count = 0; count <= n; count++)**

**{**

**/\*Below is to handle case where all elements of ar1[] are**

**smaller than smallest (or first) element of ar2[]\*/**

**if (i == n)**

**{**

**m1 = m2; m2 = ar2[0];**

**break;**

**}**

**/\*Below is to handle case where all elements of ar2[] are**

**smaller than smallest (or first) element of ar1[]\*/**

**else if (j == n)**

**{**

**m1 = m2; m2 = ar1[0];**

**break;**

**}**

**/\* equals sign because if two arrays have some common elements \*/**

**if (ar1[i] <= ar2[j])**

**{**

**m1 = m2;**

**m2 = ar1[i];   i++;**

**}**

**else**

**{**

**m1 = m2;**

**m2 = ar2[j];**

**j++;**

**}**

**}**

**return (m1 + m2)/2;**

**}**