# Problem 1: Data Types, and Operations

## Consider two hexadecimal numbers: xB3 and xC4. What decimal values do they represent for each of the five data types? (10 points)

|  |  |  |
| --- | --- | --- |
|  | xB3 | xC4 |
| Unsigned Binary | 179 | 196 |
| 1’s compliment | -76 | -59 |
| 2’s compliment | -77 | -60 |

1. **Add the following 2’s complement binary numbers. Also express the**

**answer in decimal and binary. (6 points)**

*a.* 1001+ 1010=(−13) 1 0011

*b.* 111 + 00010101=01010100=-1+21=20

*c.* 0110 + 100=0001=6-4=1

# Problem 2: Digital Logic

1. **Construct the circuit of the following truth table using AND, OR, NOT gates: (6 points)**

|  |  |  |
| --- | --- | --- |
| INPUT (A) | INPUT (B) | OUTPUT |
| 1 | 1 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 0 | 0 |

**Answer:  
A diagram of a circuit

Description automatically generated with low confidence**

1. **Write the truth table for following circuit (6 points)**

A picture containing diagram, plan, technical drawing, design

Description automatically generated

Answer:

Try again

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

# Problem 3: Von Neumann Model

1. **Explain why program counter does not always increment instruction numbers by one (5 points)**

**Answer: it may receive a branch instruction from running software telling to jump to a different point.**

1. **For a computer and terms descriptions are shown below.**

**Draw a line to connect each computer term to the best descriptions. (6 points)**

**Asnwer**:A screenshot of a computer terms

Description automatically generated with low confidence

**A diagram of a computer term

Description automatically generated with low confidence**

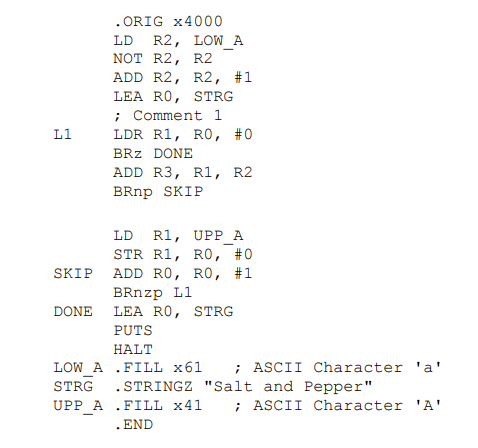
# Problem 4: LC-3 and Assembly

**How might one use a single LC-3 instruction to move the value in R3 into R4?**

**Answer:**

**ADD R4, R3,#0**

**b) Consider** **the following program written in LC-3 assembly language:**



1. Find out the symbol table: **(6 points)**

|  |  |
| --- | --- |
| Symbol | Address |
| L1 |  |
| SKIP |  |
| DONE |  |
| LOW\_A |  |
| STRG |  |
| UPP\_A |  |

A picture containing text, screenshot, font, number

Description automatically generated

1. What is the output of this program? (Please write what character will be shown in the screen) **(5 points)**

# Problem 5: Process and Disk

1. ……….. process will move to ready state when its time allocation expires **(5 points)**
   1. Blocked
   2. New
   3. Running
   4. Suspended

Answer: running

1. Time taken to switch between user and kernel mode is …….. the time taken to switch between two process. **(5 points)**
   1. More than
   2. Independent of
   3. Less than
   4. Equal to

Answer: less than

1. **What is the capacity of a disk with 2 platters, 10,000 cylinders, an average of 100 sectors per track, and 1,024 bytes per sector?: (10 points)**

**Answer would be similar to this:**

**A picture containing text, screenshot, font, line

Description automatically generated**

# Problem 6: Virtual memory

* 1. The following table shows the first 8 entries in the page map. Recall that the valid bit is 1 if the page is resident in physical memory and 0 if the page is on disk or hasn’t been allocated.

A picture containing text, font, screenshot, line

Description automatically generated

If there are 1024 bytes per page, what is the physical address corresponding to the hexadecimal virtual address 0xE72? **(10 points)**

Answer is :0XA72

# Progeramming part

# Problem 7.1: Parallel sum (5 points)

Write a C multithread program to compute the average of an array using two threads. First threads is responsible of calculating the average of the first half of array, while second thread is responsible for the average of the second half. The final average should be printed in the output.

# Problem 7.3: Ordering threads (10 points)

Single Writer, Two Readers Synchronization

You have a programming task to implement a synchronization problem involving a single writer thread and two reader threads. The writer is responsible for incrementing a counter , while the readers print the current value of the counter. Your goal is to create a program that allows the readers to work in parallel while ensuring that the writer increments the counter only when there are no active readers.

Requirements:

* Implement the program in C programming language.
* Define a global integer variable called "counter" and initialize it to zero.
* Implement a writer function that increments the counter by a specified value (e.g., 1) and prints the updated value to the console. The writer function should be executed by the writer thread.
* Implement a reader function that reads the current value of the counter and prints it to the console. The reader function should be executed by each reader thread.
* Use appropriate synchronization techniques to ensure the following:
  + Multiple reader threads can access the counter concurrently without conflicts.
  + The writer thread can increment the counter only when there are no active readers. In other words, if any reader is currently accessing the counter, the writer must wait until all readers have finished.
* Implement the synchronization mechanism using mutex (pthread\_mutex\_t) and condition variable (pthread\_cond\_t) to coordinate the activities of the threads.
* Use pthread\_create to create the threads and pthread\_join to wait for their completion.

#include <stdio.h>

#include <pthread.h>

pthread\_mutex\_t mutex;

pthread\_cond\_t readers\_cond;

int counter = 0;

int active\_readers = 0;

void\* writer(void\* arg) {

while (1) {

pthread\_mutex\_lock(&mutex);

// Wait until there are no active readers

while (active\_readers > 0) {

pthread\_cond\_wait(&readers\_cond, &mutex);

}

// Increment the counter

counter++;

printf("Writer: Counter incremented to %d\n", counter);

pthread\_mutex\_unlock(&mutex);

// Break the loop after incrementing the counter

break;

}

pthread\_exit(NULL);

}

void\* reader(void\* arg) {

pthread\_mutex\_lock(&mutex);

active\_readers++;

pthread\_mutex\_unlock(&mutex);

// Read the counter

printf("Reader: Counter value is %d\n", counter);

pthread\_mutex\_lock(&mutex);

active\_readers--;

// Signal the writer if no more active readers

if (active\_readers == 0) {

pthread\_cond\_signal(&readers\_cond);

}

pthread\_mutex\_unlock(&mutex);

pthread\_exit(NULL);

}

int main() {

pthread\_t writer\_thread, reader\_thread1, reader\_thread2;

pthread\_mutex\_init(&mutex, NULL);

pthread\_cond\_init(&readers\_cond, NULL);

// Create writer thread

pthread\_create(&writer\_thread, NULL, writer, NULL);

// Create reader threads

pthread\_create(&reader\_thread1, NULL, reader, NULL);

pthread\_create(&reader\_thread2, NULL, reader, NULL);

// Wait for threads to finish

pthread\_join(writer\_thread, NULL);

pthread\_join(reader\_thread1, NULL);

pthread\_join(reader\_thread2, NULL);

pthread\_mutex\_destroy(&mutex);

pthread\_cond\_destroy(&readers\_cond);

return 0;

}