**Student name and email:**

**Student number:**

Exam info

## Exam is written

* There are 100 points in total, which will be used to decide the grade.
* about 75% of the questions/points are written problems.
* about 25% programming exercises; here you can use the tools for development.

## Censorship

There is external censorship.

## Exam Guidelines:

* The exam must be completed at AAU campus; it will be 4 hours.
  + The first two and half hours will be the written part, and the last part will be programming.
* The exam is handed out in paper.
* The programming answers must be submitted in digital exam as a file.
* If you think that the task formulation is deficient / incorrect, then write which additional assumptions / changes you use as a basis for your answer.
* Remember to clearly write the name and study number on the answer.

### Permitted aids:

* Common IT tools such as text editor / word processing, PDF annotation tool, Calculator.
* All tools that are introducing in the course.
* Formats of the entire LC-3 instruction set (page 148 of ICS book).
* The *common.h* and *common\_threads.h* files from the OSTEP book project.

Written part

# Problem 1: Data Types, and Operations

## (5 points) Consider two hexadecimal numbers: x43 and xA4. What values do they represent for each of the five data types?

Answer:

|  |  |  |
| --- | --- | --- |
|  | x43 | xA4 |
| Unsigned Binary | 67 | 164 |
| 1’s compliment | 67 | -91 |
| 2’s compliment | 67 | -92 |

## (5 points) Calculate the following expression in two’s complement. Also express the answer in decimal.

Answer:

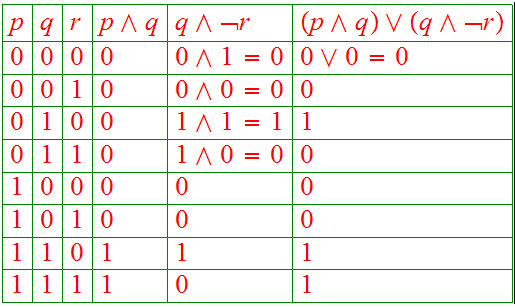
* 10101 – 00111= 10101+11001 = 101110 = -18
* 10101 – 10111 = 10101+01001 = 11110 = -2.
* 10110101 + 00111011 = 11110000 = -75 + 59 = -16

# Problem 2: Digital Logic

## (7 points) Fill in the truth table for the following logical expression.

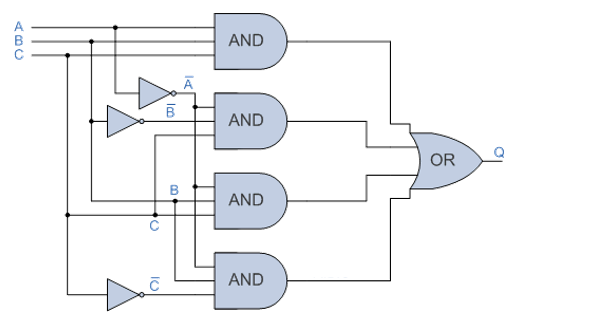
**F = ((p AND q) OR (q AND (NOT(r)))**

Answer:

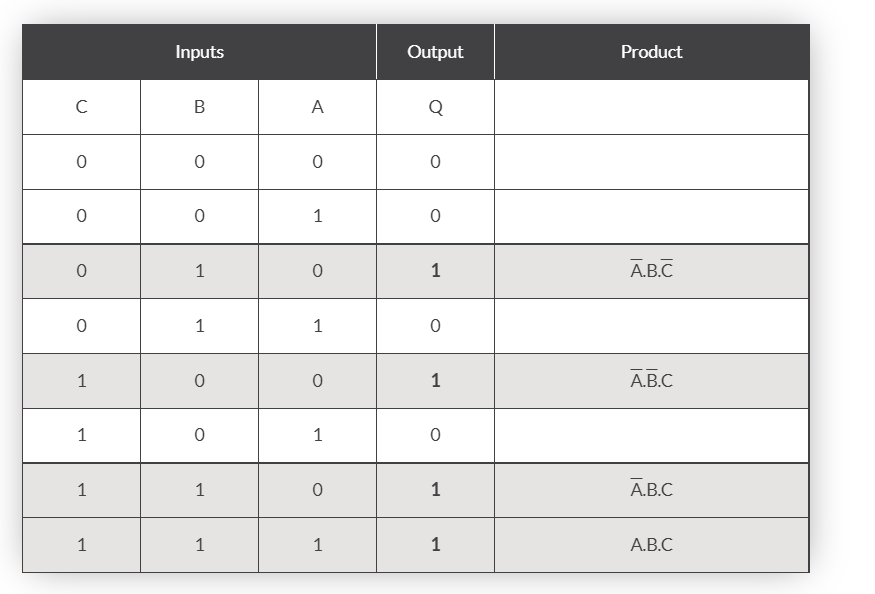


## (6 points) Construct the output truth table for the following logic circuits that consist

## of two exclusive-OR gates.



Answer:



# Problem 3: Von Neumann Model

## (6 points) The table below shows the memory from a von Neumann model counter. If the program counter contains 10010, state the data that will be placed in *Memory Address register* and in *Memory Data register*.

* State the data that will be placed in Memory Address register?

Answer: 10010

* State the data that will be placed in Memory Data register   
  Answer: 1110001

|  |  |
| --- | --- |
| Address | Contents |
| 10001 | 11001101 |
| 10010 | 11110001 |
| 10011 | 10101111 |
| 10100 | 10000110 |
| 10101 | 00011001 |
| 10110 | 10101100 |

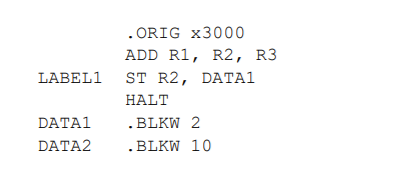
## (5 points) The six Stages in von Neumann cycle are shown in table below. Put each stage in correct sequence by writing the numbers 1 to 6 in the right column.

Answer:

|  |  |
| --- | --- |
| Stage | Sequence number |
| The instruction is then copied from the memory location contained in MAR the (Memory address register) and is placed in the MDR (Memory data register) | 3 |
| The instruction is finally decoded and is then executed | 6 |
| The PC (program counter) contains the address of the next instruction to be fetched | 1 |
| The entire instruction is then copied from MDR and placed in IR (Instruction Register) | 4 |
| The address contain in PC is copied to MAR via address bus | 2 |
| The address part of the instruction, if any, is placed to MAR | 5 |

# Problem 4: LC-3 and Assembly

## (6 points) What are the addresses corresponding to the symbols LABEL1, DATA1, DATA2 in the following code?



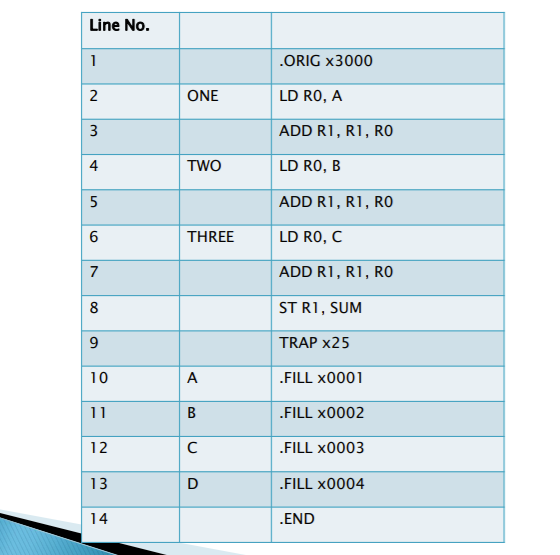
Answer:

LABEL1: x3001

DATA1: x3003

DATA2: x3005

## (4 points) The following program adds the values stored in memory locations A, B, and C, and stores the result into memory. There are two errors in the code. For each, describe the error and indicate whether it will be detected at assembly or run time.



Answer:

1. SUM is not defined – assembly time error.
2. R1 is not cleared before the ADD operation, run time error (R1 might contain a value and should be set to zero).

## (5 points) Write an assembly program to read 5 and 6 from memory and multiply them and store the result in R6.

Answer:

;

; Program to multiply a number by the constant 6

;

.ORIG x3050

LD R1, SIX

LD R2, NUMBER

AND R3, R3, #0 ; Clear R3. It will

; contain the product.

; The inner loop

;

AGAIN ADD R3, R3, R2

ADD R1, R1, #-1 ; R1 keeps track of

BRp AGAIN ; the iteration.

;

HALT

;

NUMBER .BLKW 1

SIX .FILL x0006

;

.END

# Problem 5: Process and Disk

## (5 points) Consider the set of 5 processes whose arrival time and burst time are given below. Assuming that the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average turnaround time.

|  |  |  |
| --- | --- | --- |
| Process ID | Arrival time | Burst time |
| P1 | 0 | 5 |
| P2 | 1 | 3 |
| P3 | 2 | 1 |
| P4 | 3 | 2 |
| P5 | 4 | 3 |

Answer:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P3 | P4 | P5 | P1 | P2 | P5 | P1 |
| 2 | 4 | 5 | 7 | 9 | 11 | 12 | 13 | 14 |

Turn around time for each:

P1=14

P2=11

P3=3

P4=4

P5=9

## (5 points) Consider a disk with a rotational rate of 10,000 RPM, an average seek time of 8 ms, and an average of 500 sectors per track. Estimate the average time to read a random sector from disk. Do this by summing the estimates of the seek time, rotational latency, and transfer time.

Answer:

The average access time is the sum of the seek time, rotational latency, and transfer time. The seek time is given as 8ms. Once the head is in the right place, on average we will need to wait for half a rotation of the disk for the correct sector to come under the head. Thus, on average, the rotational latency is half the time it takes the disk to make a complete revolution. The disk spins at 10000 RPM, so it takes 1/10000 of a minute to make one revolution. Equivalently, (1000 ms/sec × 60 sec/minute) / 10000 RPM = 6 ms to make one revolution. So rotational latency is 3ms. The transfer time is the time it takes for the head to read all of the sector. The head is now at the start of the sector, so how long does it take for the entire sector to go past the head? Since there are on average 500 sectors per track, we need 1/500th of a revolution of the disk to read the entire sector. We can work this out as (time for one revolution of disk) / 500 = 6ms / 500 = 0.012ms. So the total time is 8ms + 3ms + 0.012ms ≈ 11ms. We can clearly see that getting to read the first byte of the sector takes a long time, but reading the rest of the bytes in the sector is essentially free.

# Problem 6: Virtual memory

## (4 points) Assume we have a virtual memory detailed as follows:

* 256 MB Physical Address Space
* 4 GB Virtual Address Space
* 1 KiB page size
* A TLB with 4 sets that is 8-way associative with LRU replacement.

For the following questions it is fine to leave your answers as powers of 2. How many bits will be used for:

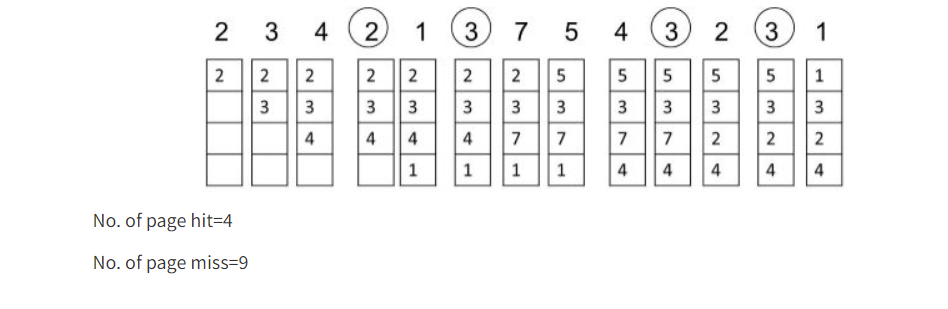
Answer:

* 1. Page offset? \_\_\_\_\_10\_\_\_\_\_\_
  2. Virtual Page Number (VPN)? \_\_\_\_22\_\_\_\_\_
  3. Physical Page Number (PPN)? \_\_\_18\_\_\_\_\_\_
  4. How many entries in this page table? 2^22

## (6 points) Considering the following page reference string, and the number of frames in the memory is 4, calculate page hit and page miss using LRU.

**2, 3, 4, 2, 1, 3, 7, 5, 4, 3, 2, 3, 1**

Answer:



## (5 points) Consider a machine with 128 MB physical memory and a 30-bit virtual address space. If the page size is 4KB, what is the approximate size of the page table?

Answer:

Take the 4 KB and how many is needed for the number of pages

4 KB =

as we need to map every possible virtual address

So, we need  entries in the page table. Physical memory being 128 MB, a physical address must be 27 bits and a page address needs 27−12=15 address bits. So, each page table entry must be at least 15 bits. So the answer .

# Problem 7: Multithreading

## (1 point) A semaphore was initialized to 10. Then 6 P (wait) operations and 4 V (post) operations were completed on this semaphore. What is the resulting value of the semaphore? Motivate your answer.

Answer:

The final value of the semaphore is 8, given that each P operation decreases the semaphore value by 1 while each V operation increases by 1.

So, the final value will be 10 – 6 + 4 = 8

## (1 point) Choose the correct answer. A critical section is a program segment:

|  |  |  |
| --- | --- | --- |
|  | A | which should run in a certain specified amount of time |
|  | B | which avoids deadlocks |
| X | C | where shared resources are accessed |
|  | D | which must be enclosed by a pair of semaphore operations, P (wait) and V (post) |

## (1 point) Choose the correct answer. To implement a mutex (lock) using a semaphore, we need to initialize the semaphore to the value:

|  |  |  |
| --- | --- | --- |
|  | A | 0 |
|  | B | -1 |
| X | C | 1 |
|  | D | Any value bigger than 0 |

## (3 points) Choose the correct option and motivate the answer.

In the pseudocode below we have the semaphore *a* initialized to 1, the function *child1to9* to be executed by 9 threads T1…T9, and the function *child10* to be executed by thread T10. Assume that the critical section in both functions accesses the same shared variables.

sem\_t a;  
sem\_init(&a, 0, 1);

child1to9() {  
 while(true) {  
 sem\_wait(&a);  
 // critical section, the same as T10  
 sem\_post(&a);  
 }  
}

child10() {  
 while(true) {  
 sem\_post(&a);  
 // critical section, the same as T1…T9  
 sem\_post(&a);  
 }  
}

What is the largest number of threads that can enter the critical section?  
Motivate your answer.

|  |  |  |
| --- | --- | --- |
|  | A | 1 |
|  | B | 2 |
|  | C | 3 |
| X | D | All the threads |

Given that T10 keeps performing only post operations, could happen that this thread gets to run for so long time that the value of the semaphore becomes > 10, which would allow all the threads to enter the CS.

Programming part

Answer: The solutions for this part are attached as separate C files.

# Problem 8: Write a multithreaded program that creates two threads T0, T1 and one shared array of 10 integers that you can initialize as you wish.

## (6 points) The thread T0 should print all the numbers >= 0 of the array while T1 should print all the negative numbers.

The format of the output should be 10 lines in the following format: <Tid> <nr>, where Tid is the thread id (0 or 1) and nr is a number from the array.

### Example

a = [4, 3, 7, -2, 4, -1, 6, -3, -8, 0]

### Possible output

0 4  
0 3  
1 -2  
1 -1  
0 7  
1 -3  
0 4  
0 6  
0 0  
1 -8

## (6 points) Create a copy of the above solution with the only difference that T0 should complete first and then T1.

### Example

a = [4, 3, 7, -2, 4, -1, 6, -3, -8, 0]

### Expected output

0 4  
0 3  
0 7  
0 4  
0 6  
0 0  
1 -2  
1 -1  
1 -3  
1 -8

# Problem 9: (8 points) Write a multithreaded program that creates 4 threads T0…T3. Each thread simply prints “Hello world <Tid>” where Tid is the ID of the thread (i.e., 0, 1, 2, or 3). Use condition variables to make so that the threads with even ID will print first and then those with odd ID will print.