COMP1562 – Operating System

Laboratory 2

Processing Programs

Group ID: 21

Group Task: Task 1

Usman Basharat

Mohamed Aden

Yunus Hassan

Derek U. Mafohla

Table of Contents

Task 1.1

1.1. The hypothetical 16 bit processor in the PowerPoint slides also has two I/O instructions:

0011 = Load AC from I/O
0101 = Add to AC (add date from the given memory location)
0111 = Store AC to I/O

Your system contains additionally two I/O device buffers at addresses 005h and 006h. Instructions should be placed in the memory locations 300h, 301h and 302h. Memory locations to store / get data are 900h, 901h and 902h. Assume initial values for I/O buffers 005h and 006h to be, respectively, 0001h and 0001h and initial values in the memory locations to be, respectively, 0100h, 0010h and 0001h. All registers values (PC, accumulator and IR, accordingly to the diagram for pseudo code execution) should be updated accordingly. There are three steps to be processed (each of the steps comprises of fetch and execute part):

a. Load AC from the device buffer at address 005h.
b. Add contents of memory location 901h.
c. Store AC to device buffer at address 006h.

Step 1:				Step 2:			
Memory:		CPU Registers:		Memory:		CPU Registers:	
300:	3005	PC:	300	300:	3005	PC:	301
301:	5901	AC:		301:	5901	AC:	0001
302:	7006	IR:	3005	302:	7006	IR:	3005
900:	0100			900:	0100		
901:	0010			901:	0010		
902:	0001			902:	0001		
I/O:				I/O:			
005:	0001			005:	0001		
006:	0001			006:	0001		

During this, we start by layout each stage of the process. This is used by putting in the correct values in this stage. Stage one consists of the program counter being started by using the first memory location and instruction register is enabling this in the instruction register. Accumulator is empty, because no instruction has been stored yet. Stage 2 is the part where it stores the instruction register by storing the memory by adding 1 to this into the accumulator. The program counter for stage 2 goes up to one because one instruction has been executed. This would be the same for the next stages to come too.

Step 3:

Memory:		CPU Registers:	
300:	3005	PC:	301
301:	5901	AC:	0001
302:	7006	IR:	5901
900:	0100		
901:	0010		
902:	0001		
I/O:			
005:	0001		
006:	0001		

Step 4:

Memory:		CPU Registers:	
300:	3005	PC:	302
301:	5901	AC:	0011
302:	7006	IR:	5901
900:	0100		
901:	0010		
902:	0001		
I/O:			
005:	0001		
006:	0001		

Stage 3 is the next part of the process. Stage 3 and 4 consists of storing the next memory location which is 5901. This is done the same as Stage 2 and 3. Adding 5901 to the instruction register and for stage 4, is again, adding 1 to the instruction to know that this has been done. The program counter for stage 4 goes up to one because one instruction has been executed. Stage 3 is the same, because the accumulator has not gone one up to indicate that the instruction has not yet been stored.

Step 5:

Memory:		CPU Registers:	
300:	3005	PC:	302
301:	5901	AC:	0011
302:	7006	IR:	7006
900:	0100		
901:	0010		
902:	0001		
I/O:			
005:	0001		
006:	0001		

Step 6:

Memory:		CPU Registers:	
300:	3005	PC:	303
301:	5901	AC:	0011
302:	7006	IR:	7006
900:	0100		
901:	0010		
902:	0001		
I/O:			
005:	0001		
006:	0011		

Stage 5 is the same as Stage 3. This means that the next instruction, which in this case, is 302 that is stored in the instruction register. However, for stage 6, it does not accumulate, because it needs to store the accumulator into the output, which is stored in 006. At this stage, it realises that everything is complete, and it stores it into the output. The only similarity between stage 6 and 4 is the program counter going up again.

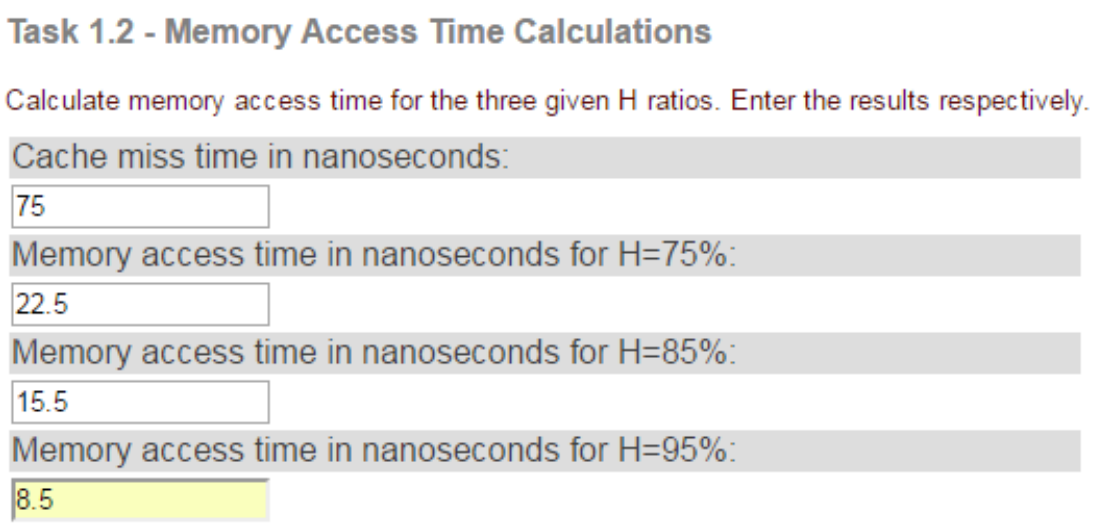
Task 1.2

1.2. Consider a machine with:

a. 1k words cache, access time 5ns.
b. 1M words memory, access time 70ns.
c. If the data is not in cache then the data is copied from memory.

Calculate (H is cache success ratio):

a. Cache miss time.
b. data access time for H=75%
c. data access time for H=85%
d. data access time for H=95%



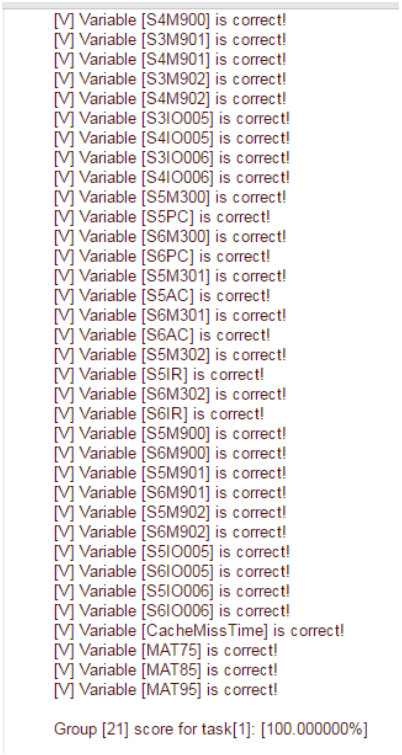


Figure 1 shows the result of these tasks

Cash miss time is calculated by adding the cache access time, which is 5ns, and the memory access time, which is 70ns. By adding these two together, we get 75nanoseconds which is the final answer for the cache miss time. Memory access time for 75% is calculated by 0.75*5+0.25*(70+5) = 22.5. Memory access time for 85% is calculated by 0.85*5+0.15*(70+5) = 15.5. Memory access time for 95% is calculated by 0.95*5+0.05*(70+5) = 8.5.

Figure 1 shows the result we received for this lab which is 100%.

Reflection

During the tasks for this laboratory, I found these tasks pretty easy as soon as I found out the figures of Task 1.1. I felt that by using the lecture notes from the previous week was a huge step stone to the answers that I have currently got from them. Task 1.2 was easy too, because an example of the calculations were given in the lecture notes. I found that by using the examples on the lecture notes for both of the examples was huge. Looking back at these tasks, I felt that this has given me knowledge of how registers go amount with communicating with each other of storing values within the specific locations and how to calculate the cache time.