FETCH DECODE EXECUTE

PART A

The unit page gives us some information about volatile memory, non-volatile memory and translators. Answer the following questions.

1. List 3 types of non-volatile (secondary) memory
    -
    -
    -
2. Identify a difference between SSD and HDD
3. List 3 types of primary memory
    -
    -
    -
4. Find an image of a Network Attached Storage (NAS) device and paste it here:

5. Name two types of translators:
    -
    -

6. Explain the purpose of a translator

7. Explain the difference between the two types of translators identified in Q5.

PART B

The Fetch-Decode-Execute cycle is used by your CPU to:

- Fetch an instruction from RAM
- Decode the instruction ie figure out what to do
- Execute the instruction

Here is a sample program that the CPU will attempt to run:

```
None
LOAD 32
ADD 33
STORE result
DISPLAY result
```

Look at this instruction. It has 2 parts. Which is the **Operand** and which is the **OpCode**?

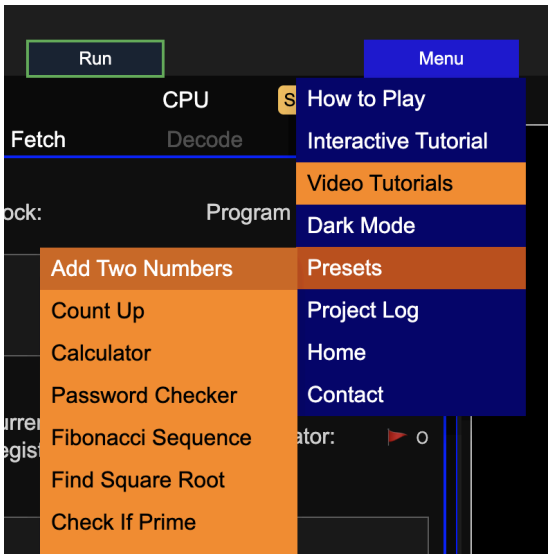
Instruction	OpCode	Operand
LOAD 32[1]		

But load it where? And how? Let's find out.

[1] *LOAD 32 means load the value in RAM memory address 32 into the CPU*

Task 1

Open the simulator. Find *Menu->Presets->Add Two Numbers*



Also, click 

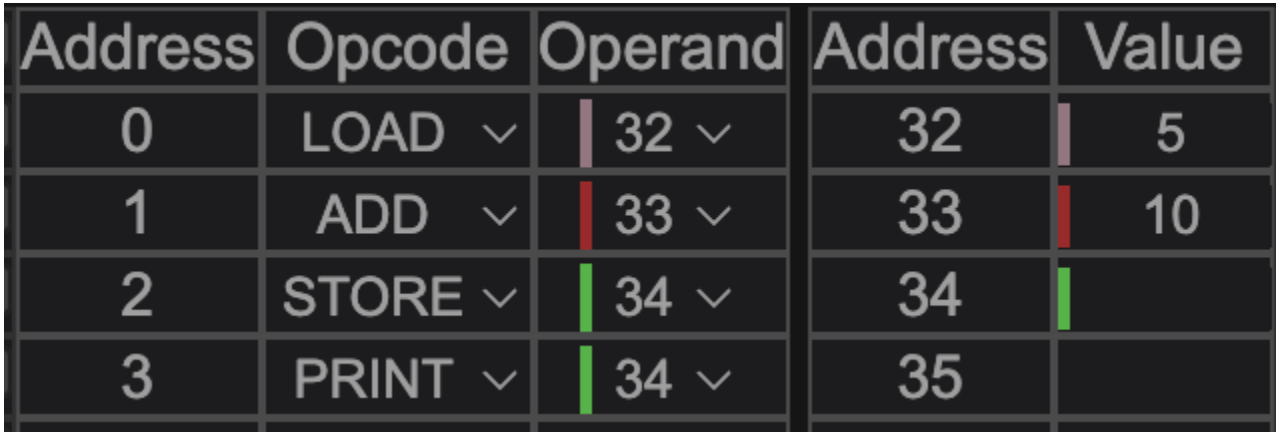
You should now see a lot of different registers.

These registers are just small blocks of memory that hold data temporarily as the cycle is in progress.

We need to understand the purpose of each register in the cycle.

Task 2

Click the Assemble button to load the program into RAM:



Address	Opcode	Operand	Address	Value
0	LOAD	32	32	5
1	ADD	33	33	10
2	STORE	34	34	
3	PRINT	34	35	

The address of the first instruction is 0.

Write down the 4 addresses in RAM of the locations that store instructions:

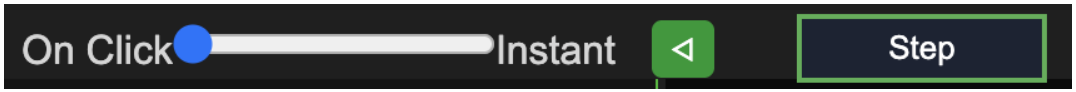
0			

Write down the 2 addresses in RAM of the locations that store data:

Task 3

In the CPU, the Program Counter (PC) will store the address of the first instruction. This is the starting point of executing the program.

Set the speed of the simulation to the lowest state:



Now we can step through the simulation at our own pace. Also, note there will be some commentary in the **console** in the bottom right corner.

Task 4

Click 

and analyse what is going on.

You can always click 

to start again.

Check the correct answer.

The address of the next instruction to be fetched from RAM is copied from the PC to the:	MAR ▾
The instruction at that address is fetched from RAM and placed in the:	MAR ▾

It is then loaded into the:	MAR
Once the instruction has been successfully fetched from RAM, which register is updated before the DECODE process happens?	MAR
FETCH COMPLETE	
Now the instruction in the CIR needs to be **decoded** by the CU (control unit). **LOAD 32** means we need to load the contents of address 32 in RAM.	
DECODE COMPLETE	
And so, to start **executing** the instruction, the address 32 is placed in the MAR. This address will be located in RAM. Which value will be returned from RAM and where will it be stored at first?	
Because a value (not an instruction) is being returned from RAM, where will the value be sent next?	
EXECUTE COMPLETE	
The first instruction at memory location 0 has been fetched, decoded and executed! Hopefully	⭐⭐⭐⭐⭐

you can see that the value 5 has been loaded into the ACC (accumulator). Rate your understanding of the process so far!	
Oh! Remember at the end of the FETCH cycle, the PC was updated! So, now we are ready to FETCH the next instruction at address 1 in RAM. Where will the address in the PC be copied to?	MAR
The address stored in MAR will be located in RAM. Where will the instruction at that address be sent first?	MAR
Where will it then be sent? *Note, it is an instruction, not a value.	MAR
END OF FETCH! But wait - a register needs to be updated! Which one?	MAR
START OF DECODE - the instruction in CIR is **ADD 33**. The control unit analyses the OpCode and Operand and determines what needs to be done. Because arithmetic will be undertaken, where does the control unit send the **value** currently stored in the ACC?	
START OF EXECUTE: The address 33 is sent to	MAR

This address is located in RAM. Which register is first used to store the **value** returned from that address ?	MAR
Where is this **value** then sent?	MAR
Now that we have 2 values, one in the ACC and one on the ALU, the ALU can do the arithmetic! The result of the arithmetic is sent to:	MAR
END OF EXECUTE	
START OF FETCH....	

Using the simulator, can you figure out the steps that will be taken in order to complete the program. You do not have to write the steps down but you can if you want. It might be helpful to find a partner and go through the steps together. Discussion will help you figure out the FETCH-DECODE-EXECUTE cycle.

TASK 5

Now that we have some understanding of the different registers used in the FETCH-DECODE-EXECUTE cycle, complete the following table. You can do further research but you do not need to go into great depth.

Register Name	Purpose
PC (program counter)	Stores the address of the next instruction to be fetched from RAM
MAR (memory address register)	
MDR (memory data register)	
CIR (current instruction register)	
ACC (accumulator)	
CU (control unit)	
ALU (arithmetic logic unit)	

TASK 6

Go through the simulation one more time. This time turn 

It mentions L1, L2, L3 cache. The **console** also discusses cache as you step through the simulation.
What is cache and what is the difference between L1, L2 and L3 cache?

Task 7

The terms ***Cache Hit*** and ***Cache Miss*** are often used.

The CPU will first look in Cache for data that it needs. If it finds the data, this is called a **Cache Hit**!

If it can't find the data in Cache, a **Cache Miss**, then it will have to look for the data in RAM (main memory). This is slower than accessing data from cache.

To minimize cache misses, the CPU loads frequently used data and instructions into cache.

Exam style question:

Define the terms cache hit and cache miss. [2]

Task 8

There are 3 buses that help transfer data and instructions to and from the CPU and RAM:

Do some research and create a definition for each one:

address bus	
data bus	
control bus	