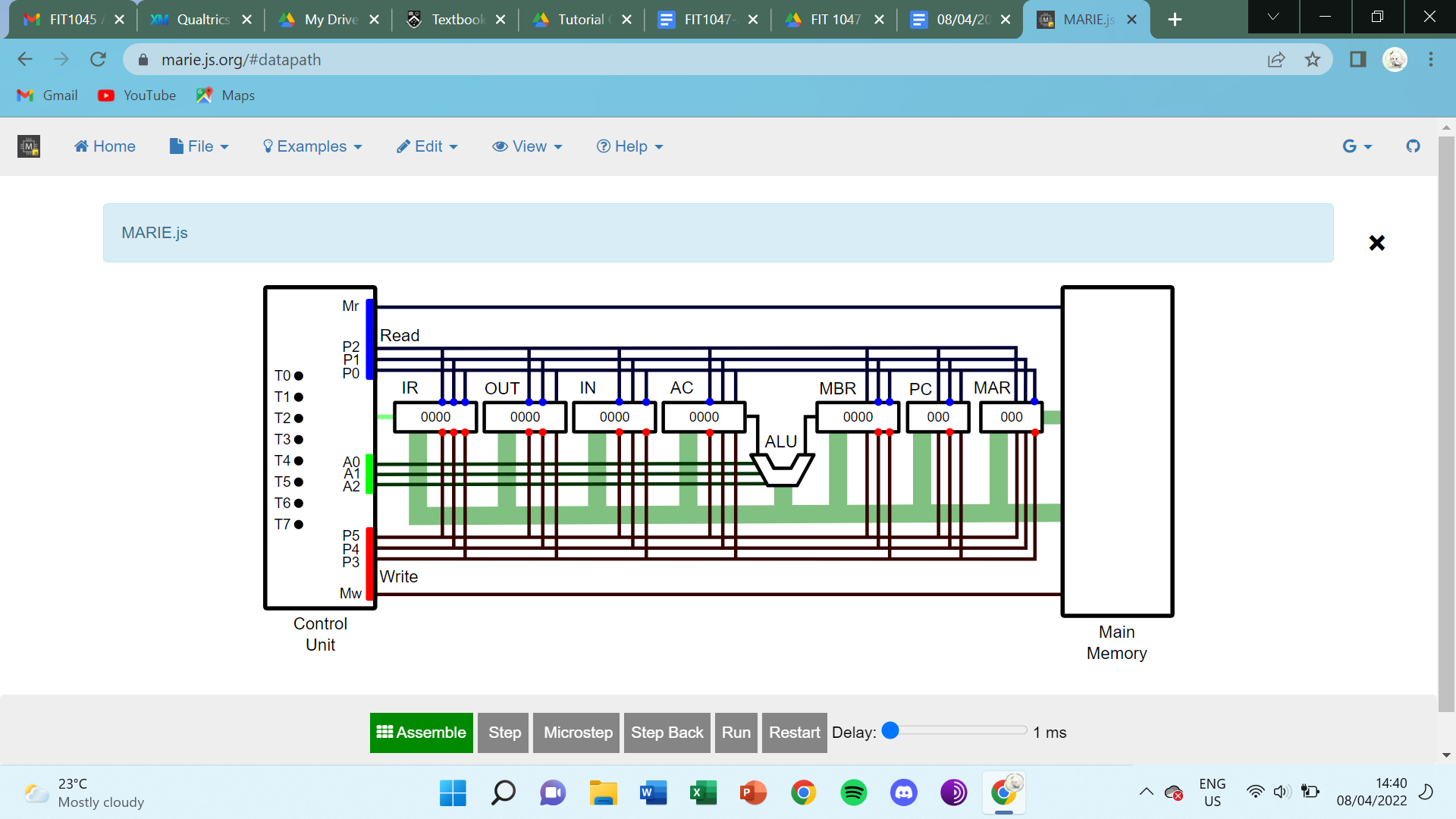
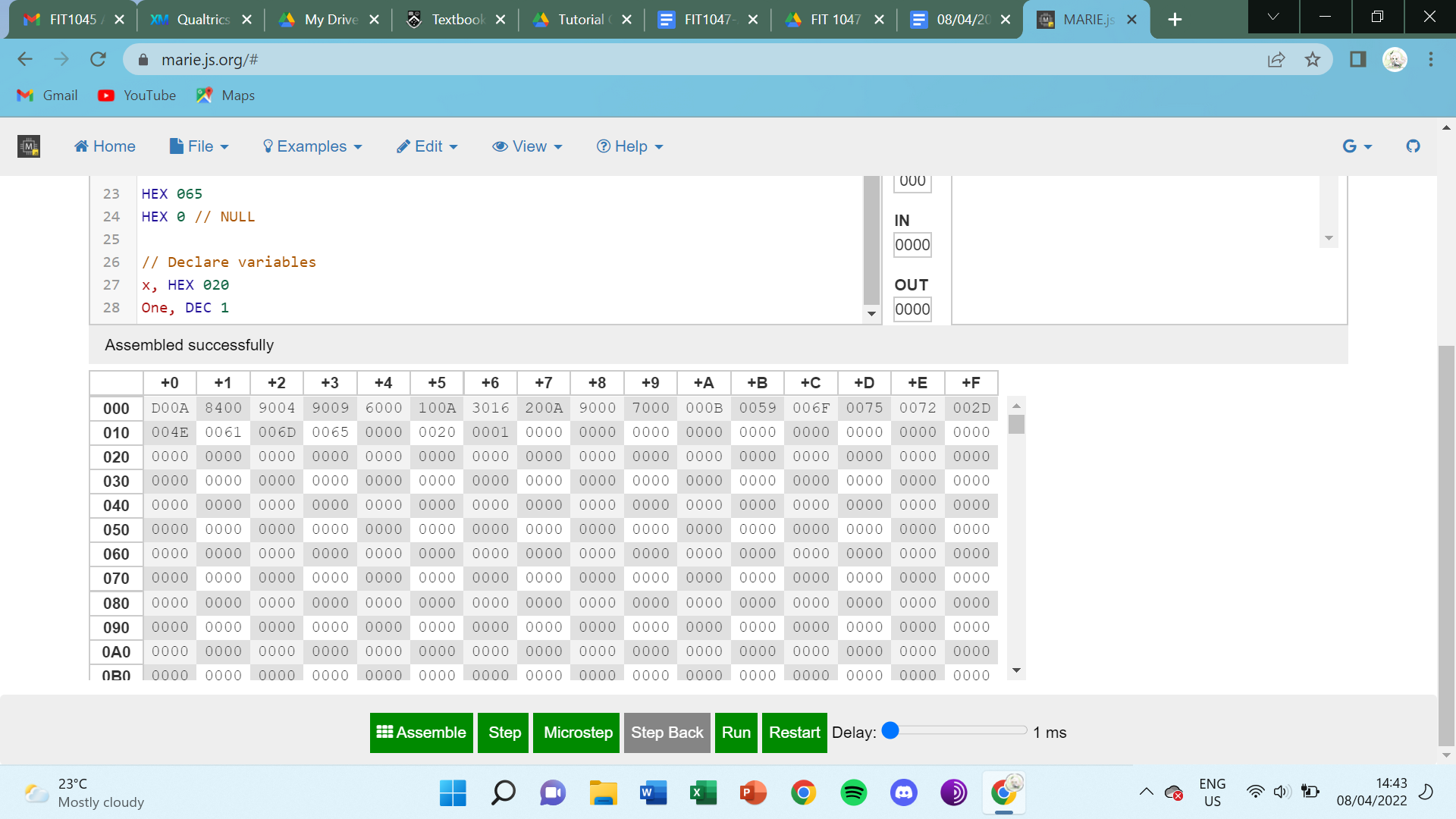
Data path can be seen on MARIE website at view → data path  
Eg. Data path image example:

Memory map used = memory below used like :

Port-mapped = allocate extra registers

Advantages:

* No one can access I/O data = no one can accidentally access the I/O data, don’t share memory
* Won’t reduce usable memory space

Disadvantages:

* costly (need more registers)
* More complex ISA (require extra instructions)

Memory-mapped =

Advantages:

* Less complex (don’t require extra instructions)
* Cheaper (don’t require extra registers)

Disadvantages:

* Will reduce usable memory space
* Users may accidentally access into the I/O data

Booting process: (NORMALLY DONE WITHIN 2-3 SECONDS)

1. Power on → Motherboard check power good signal (check if the power is stable for certain time, enough to supply to other computer components {THERE IS A CERTAIN THRESHOLD})
   1. If fan does not start, means power have problem
2. Send signal to CPU → Clock is started → CPU starts
3. CPU will load BIOS from ROM (BIOS is located in ROM, it can’t be changed/uninstalled but can be updated)
4. POST (check all essential hardware, eg. check RAM by sending signal to RAM and RAM will respond back to CPU which signified that RAM is working) → Load basic drivers (basic drivers are needed to function certain hardware)
5. BIOS will search for bootable drive to find bootloader (bootloader contains info about computer OS)
6. Bootloader starts OS (operating system) and then user can open application and programs

BIOS load essential drivers; OS loads non-essential drivers

Activity 3(a)

| For A Round Robin scheduling process, the queue condition is given below where the jobs are to be executed giving an equal time slice to each process maintaining the order of the queue. Assuming the time quantum is 2 time-unit (time slice =2), calculate the turnaround time for the following processes.   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |   Red signify stop/end |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

NO NEED WRITE OWN SUBROUTINE, ITS GIVEN IN TEMPLATE ALREADY

MAIN PROGRAM IS WRITTEN DY

NESTED SUBROUTINE IS IMPLEMENTED DY = JUMP LAI JUMP QU

2.1 NO NEED TESTCASE

2.2 HAVE TESTCASE

TEMPLATE HAVE PLACE/POSITION TO PUT IN CODE

DEFINE RANGE, UPPER / LOWER CASE, WITHIN OR OUT RANGE, CONVERSION

ASSUME USER ONLY KEY IN UPPER OR LOWER CASE, NO ?!)\*&#$%^&\*  
FIRST TEST-CASE HAVE BEEN SHOWN LMAO, JUST HARDCODE NAME IN

COVERT THEN PRINT-OUT ^^^

OTHER TEST-CASE CANNOT HARDCODE AHHHHH

STORE AT WHERE? THE INPUT.

DEFINE A PLACE WITH A LABEL. FIRST ADDRESS TO STORE IS SMTG ToUpperWord (FIND IN PDF)

USE 10-15

2.2

SUBROUTINE LOOP THROUGH CHARACTER ONE BY ONE TO MAKE IT UPPER FROM LOWER, IF ALREADY UPPER THEN IGNORE

TEMPLATE CAN MODIFY THE CODE INSIDE

COMMENT IS BACKSLASH :’]

CAN MODIFY BUT DUN CHANGE INPUT AND PRINT

NEED LINK THEM ACCORDINGLY, DON’T JUST LIKE COPY PASTE IT THERE LMAO

CAN SEPERATE THEM FIRST, SEQUENCE TRY DON’T CHANGE, IF SEPARATE, PUT THEM BACK

TR WILL CHECK FINAL WORDLE THINGY

2.5 IS LOOPING

2.5 IS GAME

LAST TASK SCREENSHOT TESTCASE

MAYBE NO TIME FOR INTERVIEW BUT NOT SURE YET

TASK MANAGER

EXPLAIN TAB ALL THAT SHIT

PICK BACKGROUND PROCESS

ONLINE SEARCH LAHHH

IN ORDER TO MAINTAIN STUFF

NO RIGHT OR WRONG BUT DON’T SIMPLY DO

FILE MANAGEMENT, HARDDISK IS A HARDWARE, CANNOT SIMPLY ACCESS I/O ETCETC

FINISH UP TO 2.3 BY THIS WEEKEND

2 HRS TUES FOR ASSIGNMENT 2 (ZOOM SESSION FOR QUESTION)

// **Template for assignment 2**

//

// **Your assignment submission needs to be based on this template**.

//

// **Carefully read the instructions**.

//

// **You need to complete the subroutines and the test cases below.**

//

// **Make sure all code is your own, and don't share any code**

// **or code fragments with other students.**

/ Execute all the test cases

/ (You can comment the next line if you don't want to

/ run the test cases)

JnS TestAll

/ Execute the main wordle game

JnS Wordle

/ Finish execution

Halt

////////////////////////////////

/ Task 2.2: ToUpper subroutine

////////////////////////////////

/ ... insert your code here

///////////////////////////////////////

/ Task 2.3: StringContains subroutine

///////////////////////////////////////

/ ... insert your code here

///////////////////////////////////

/ Task 2.4: CheckGuess subroutine

///////////////////////////////////

/ ... insert your code here

/////////////////////////////////////////

/ Task 2.5: Main Wordle game subroutine

/////////////////////////////////////////

Wordle, HEX 0 / Return address

/ ... insert your code here

JumpI Wordle / return

/ Task 2.1: Your name as a string

Name, HEX 0

//////////////////////////

/ Pre-defined Subroutines

//////////////////////////

/ InputString subroutine

/ Expects the starting address of an area in memory

/ where the string can be stored in argument InputTo.

/ Requests a string character by character from the user.

/ When the user enters value 0 (not the character "0" but

/ the integer 0), the input is finished and the

/ subroutine returns.

/ Careful: the subroutine will overwrite any memory

/ starting from the InputTo address.

InputTo, HEX 0 / Start address where to store the string

InputString, HEX 0 / Return address placeholder

InputLoop, Input

StoreI InputTo

Skipcond 800

JumpI InputString / Return if entered value is 0

Load InputTo

Add One

Store InputTo

Jump InputLoop

/ PrintString subroutine

/ Expects the start address of a string to be printed

/ in argument PrintFrom.

/ Will print the string character by character until

/ a 0 is encountered, and then return.

PrintFrom, HEX 0 / Address to start printing from

PrintString, HEX 0 / Return address placeholder

PrintLoop, LoadI PrintFrom

Skipcond 800

JumpI PrintString / Return if character is 0

Output

Load PrintFrom

Add One

Store PrintFrom

Jump PrintLoop

One, Dec 1

/////////////////

/ Test cases

/////////////////

/ Each test case is a subroutine that requires no arguments.

/ The TestAll subroutine calls all the test cases in a row.

/ Add your own test cases as subroutines, and add the necessary

/ calls to the TestAll subroutine.

/ You can then execute individual test cases by calling them

/ (at the top of the file), or execute all of them by calling

/ TestAll (see top of the file).

TestAll, HEX 0 / Return address

JnS TestToUpper1 / Run first test case

JnS TestStringContains1 / Run second test case

JnS TestCheckGuess1 / Run another test case

JumpI TestAll / Return

/ Test case 1 for ToUpper

/ Expected output: your name in upper case

NameAdr, ADR Name

TestToUpper1, HEX 0 / Return address

Load NameAdr

Store ToUpperWord

JnS ToUpper

Load NameAdr

Store PrintFrom

Jns PrintString

JumpI TestToUpper1

/ Test case 1 for StringContains

/ Expected output: 1 if your name contains the letter F, 0 otherwise

TestStringContains1Char, HEX 46

TestStringContains1, HEX 0 / Return address

Load NameAdr

Store StringContainsStart

Load TestStringContains1Char

Store StringContainsChar

JnS StringContains

Load StringContainsResult

Output

JumpI TestStringContains1 / Return

GreatAdr, ADR Great

Great, HEX 47

HEX 52

HEX 45

HEX 41

HEX 54

HEX 0

WordsAdr, ADR Words

Words, HEX 57

HEX 4F

HEX 52

HEX 44

HEX 53

HEX 0

SelectedWord, Adr Words

/ Test case 1 for CheckGuess

/ Expected output: \_r\_\_\_

TestCheckGuess1, HEX 0 / Return address

Load GreatAdr

Store CheckGuessedWord

Load WordsAdr

Store CheckSelectedWord

JnS CheckGuess

JumpI TestCheckGuess1 / Return

**ASSIGNMENT 3 YEOROBUNNNN**

Data convert to signal to transfer data

5GB-Hertz - faster speed, close to router choose this for high speed

2.4GB-Hertz - larger coverage, cover wider range

higher bandwidth = higher bit-width (?) = better

<https://docs.google.com/document/d/1Verb_dLI9iTwJRAmcm-qvZSUVmm4cGKpxSo4CnALQOk/edit?usp=sharing>

^^^ WEEK\_7 APPLIED **IMPORTANT FOR A3**

IF bit rate high, it doesn’t mean latency low

Bit rate = how fast user receive data from someone along the channel

Higher bit rate = send faster

How 100mb is distributed…

Depend on 2 factors: propagation speed (usually fixed 2x10^8 m/s ) and distance

In sharkie, if u click like let’s say HTTP, the one highlighted in blue is the message

Extra bytes is header

UDP - SHORT MESSAGE (simpler protocol)

TCP - LONG MESSAGE

**2.2 now:**

Begin, loadI myName

skipcond 400

Jump Check

Jump End

toUpperWord, HEX 0 / was this instead of myName lol

Check, loadI myName

Subt thirtytwo

Skipcond 000

/ Add thirtytwo

JnS toUpper / JnS = Store the PC at address X and jump to X+1

/ JumpI toUpper / JumpI = Use the value at X as the address to jump to

Load myName

Add One

StoreI myName

Add thirtytwo

Store myName

Jump Check

toUpper, Subt thirtytwo

Output

Load myName

Add One

Store myName

Jump Check

End, Halt

myName, ADR Name

Name, HEX 046

HEX 06F

HEX 06F

HEX 04B

HEX 061

HEX 069

HEX 059

HEX 061

HEX 06E

HEX 03F

HEX 000

// Declare variables

One, DEC 1

thirtytwo, DEC 32

// output : &OO+AI9AN9AN