CSC 3210

Computer Organization and Programming

Lab Work 3

Dr. Zulkar Nine

mnine@gsu.edu

Georgia State University

Summer 2021

Lab Work 3 Instructions

- Lab 3(a): Load and build an existing project (3 points)
- Lab 3(b): Debug the Project (5 points)
- Lab 3(c): Math Problems (2 points)

Due Date: Posted on iCollege

Disclaimer

- The process shown in these slides might not work in every single computer due to Operating system version, Microsoft Visual Studio versions and everything.
- If you find any unusual error, you can inform the instructor.
- Instructor will help you resolve the issue.

Attendance!

Lab3(a)
Load and Build an existing project

in Microsoft Visual Studio

Lab 3(a) Instructions

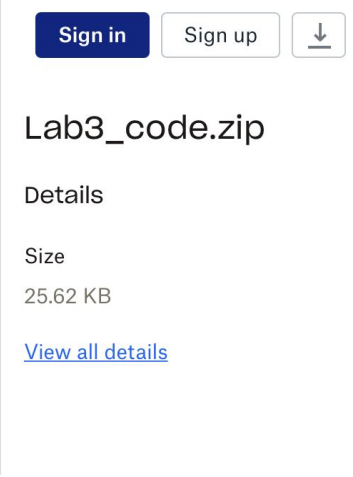
- Follow the instructions to build the project.
- Taka screenshot/screenshots showing the code, and the output window. The output window must show that the project was built successfully.
- Submit to the iCollege.

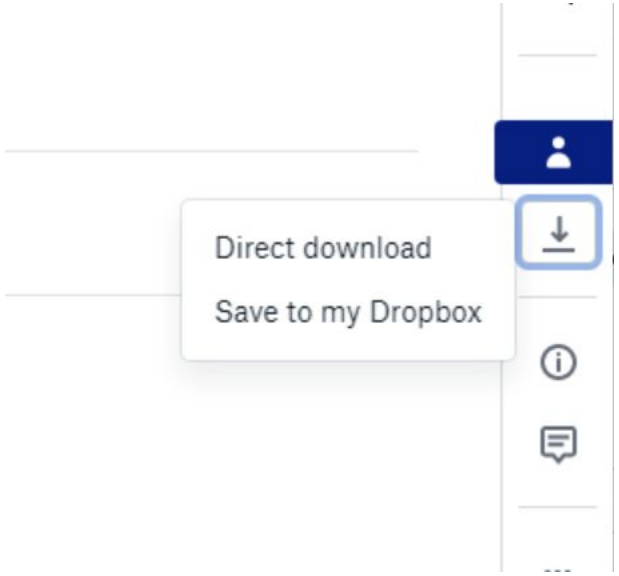
Load an existing project

- In this lab, you load an existing project.
- Follow the steps :
  - Step 1: Download the project
  - Step 2: Load the project into Microsoft Visual Studio
  - Step 3: Build the project

Step 1: Download the Project (1)

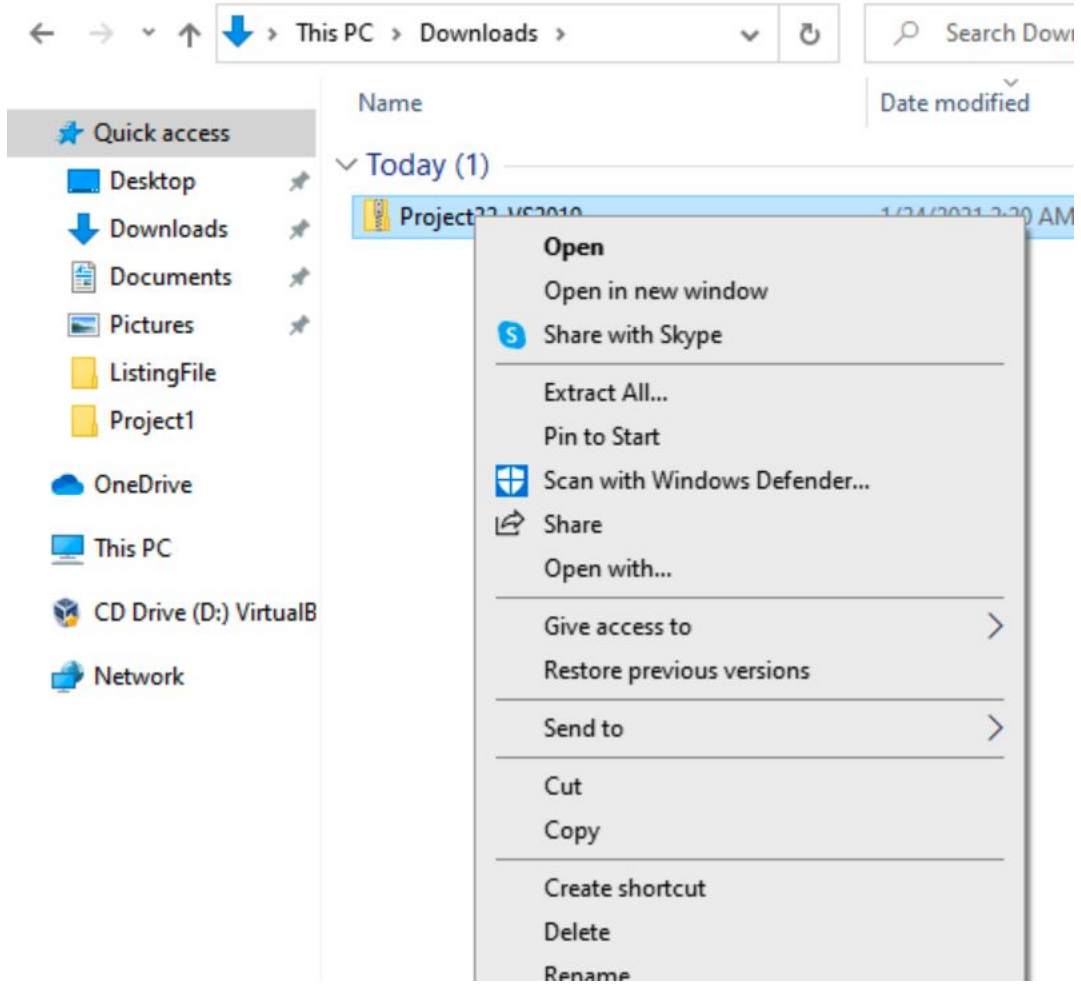
- Now download a project from the link:
- https://tinyurl.com/lab3codefall21
- Click Direct Download
- Most likely it is downloaded in your 'Downloads' folder.





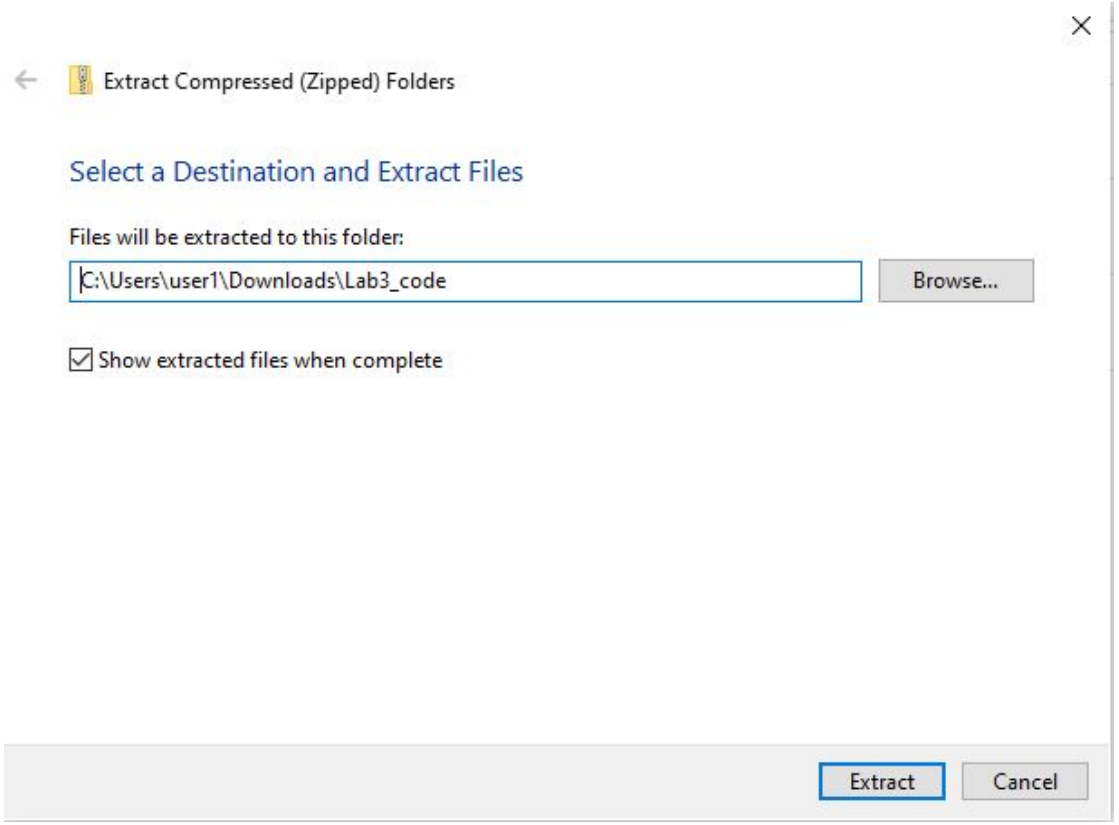
Step 1: Download the project (2)

- Go to the folder where it is downloaded
- Unzip the downloaded file
  - Right click on the zip file
  - Click Extract All…



Step 1: Download the project (3)

Remove “Lab3_code” from the path.
(keep the forward slash at the end)



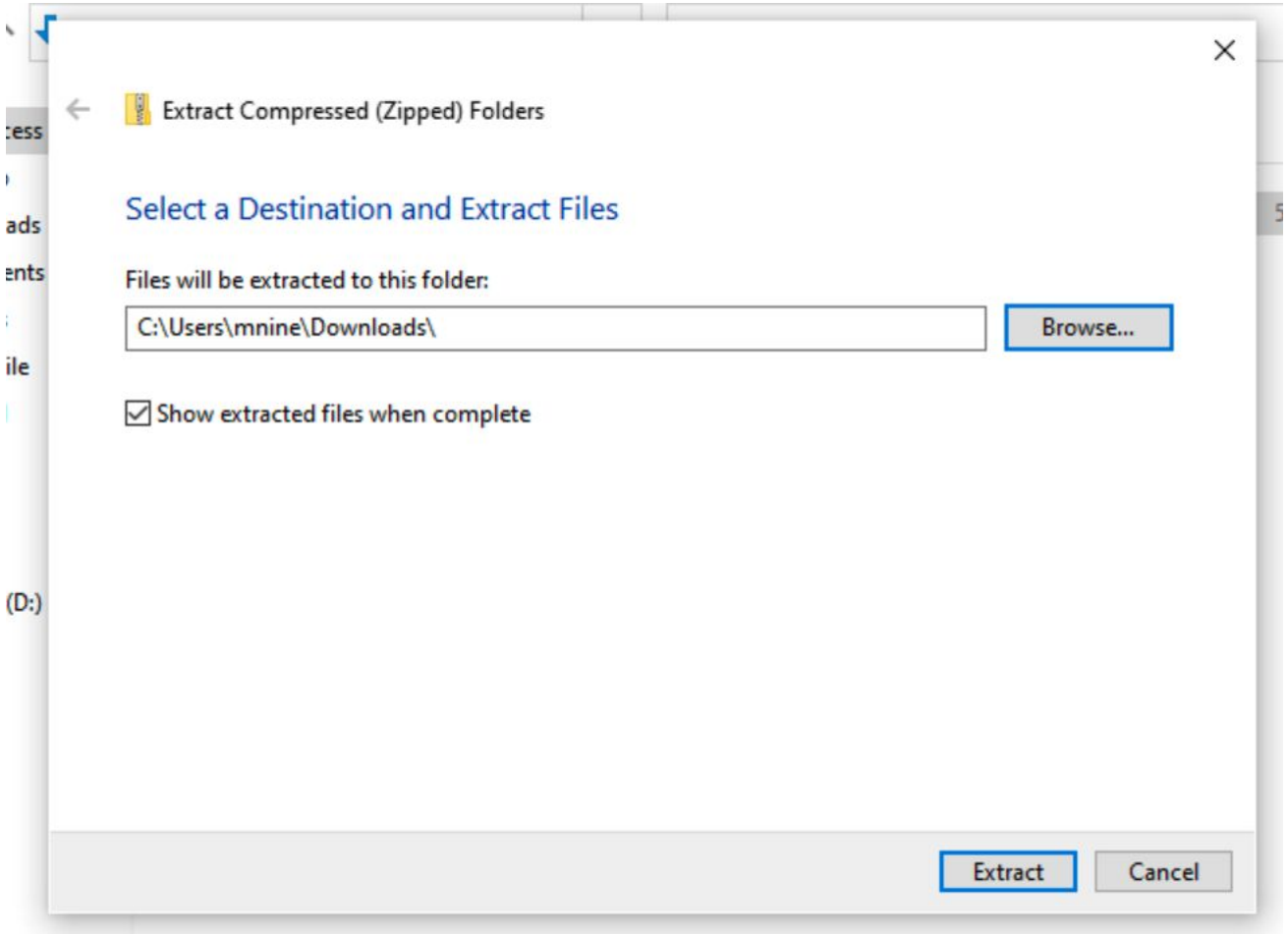
Step 1: Download the project (4)

- The path should look like
- Click Extract button
- It will extract the files in

  a folder named –

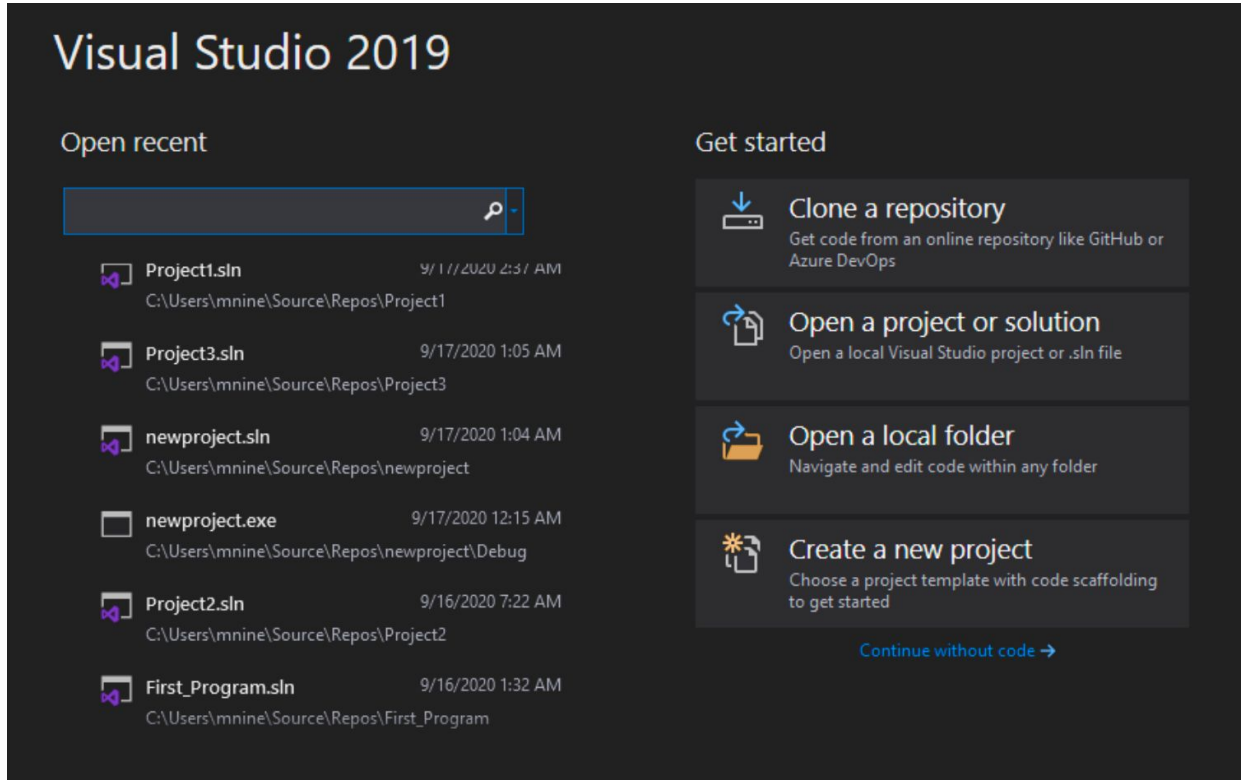
  Lab3_code

in your current directory.



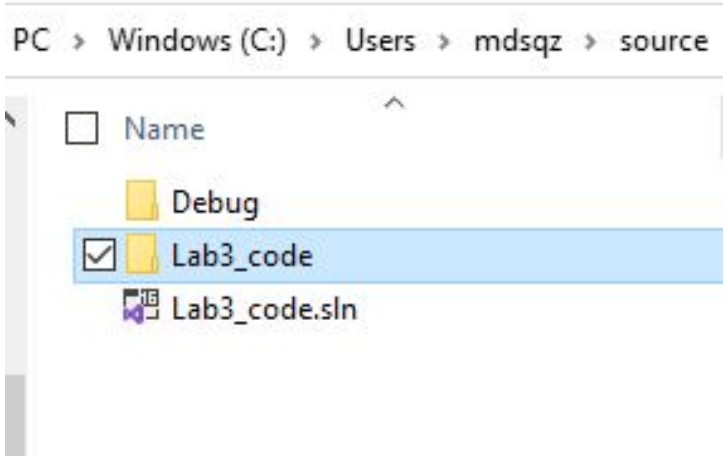
Step 2: Load the project into Visual Studio (1)

- Open Microsoft Visual Studio
- Click "Open a project or solution"



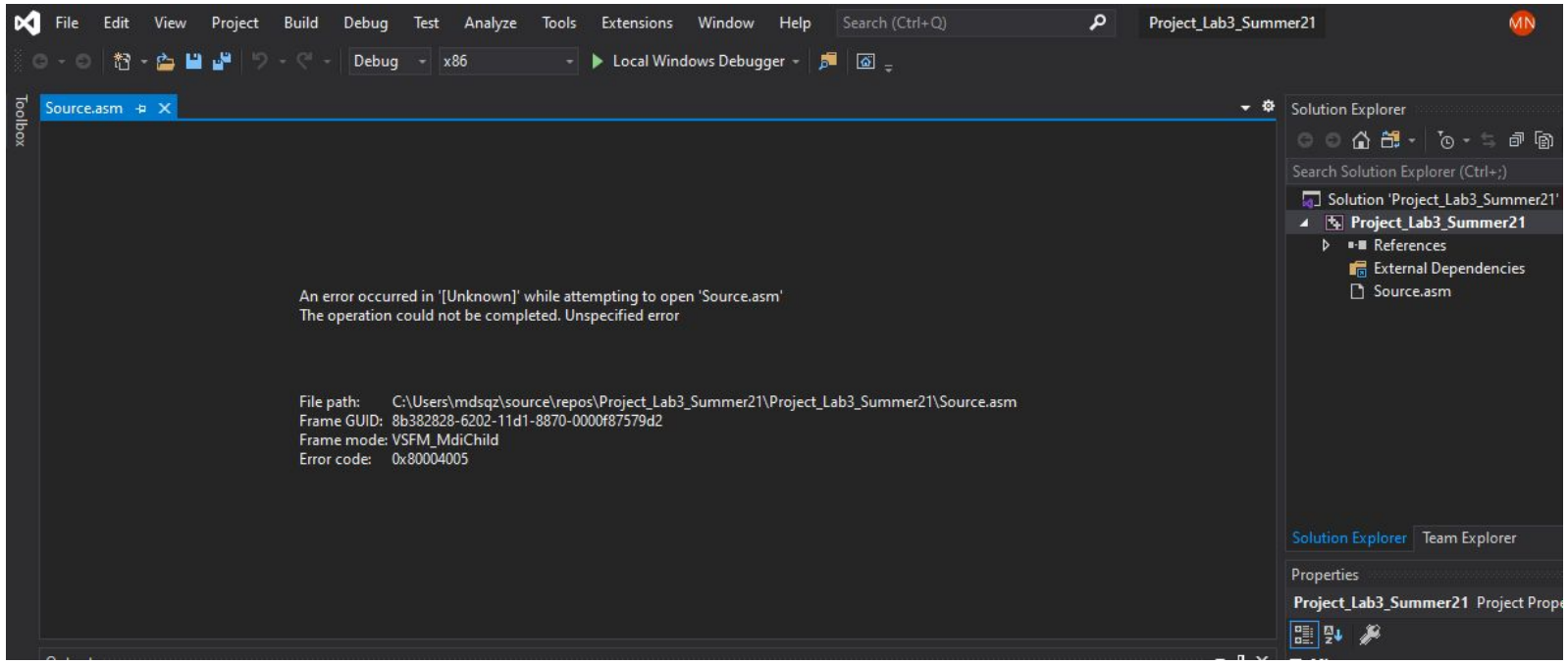
Step 2: Load the project into Visual Studio (2)

- Browse to the directory where you saved the extracted project "Lab3_code".
- Go inside the folder "Lab3_code".
- Select 'Lab3_code.sln' file.
- Click Open button at the bottom right corner.
- You might see a security warning.
- If you see one : click OK.
- The project will be loaded in you Microsoft Visual Studio.



Step 2: Load the project into Visual Studio (2)

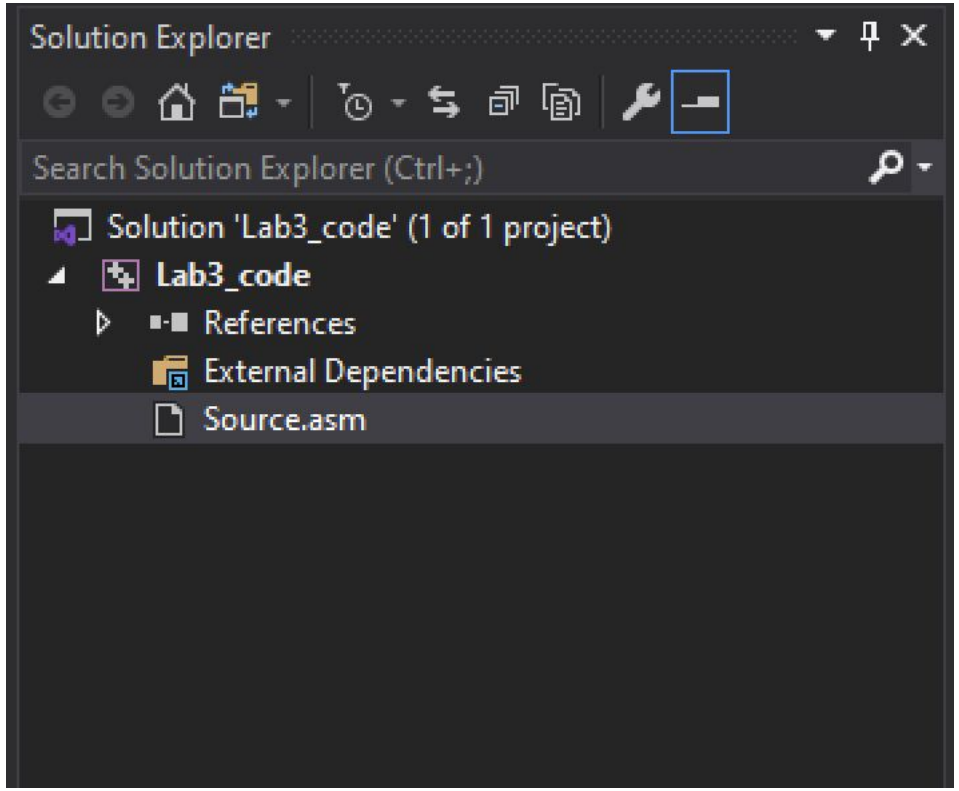
If you see the following error!



Close the Source.asm file by clicking ‘X’
Then follow the next slide

Step 2: Load the project into Visual Studio (3)

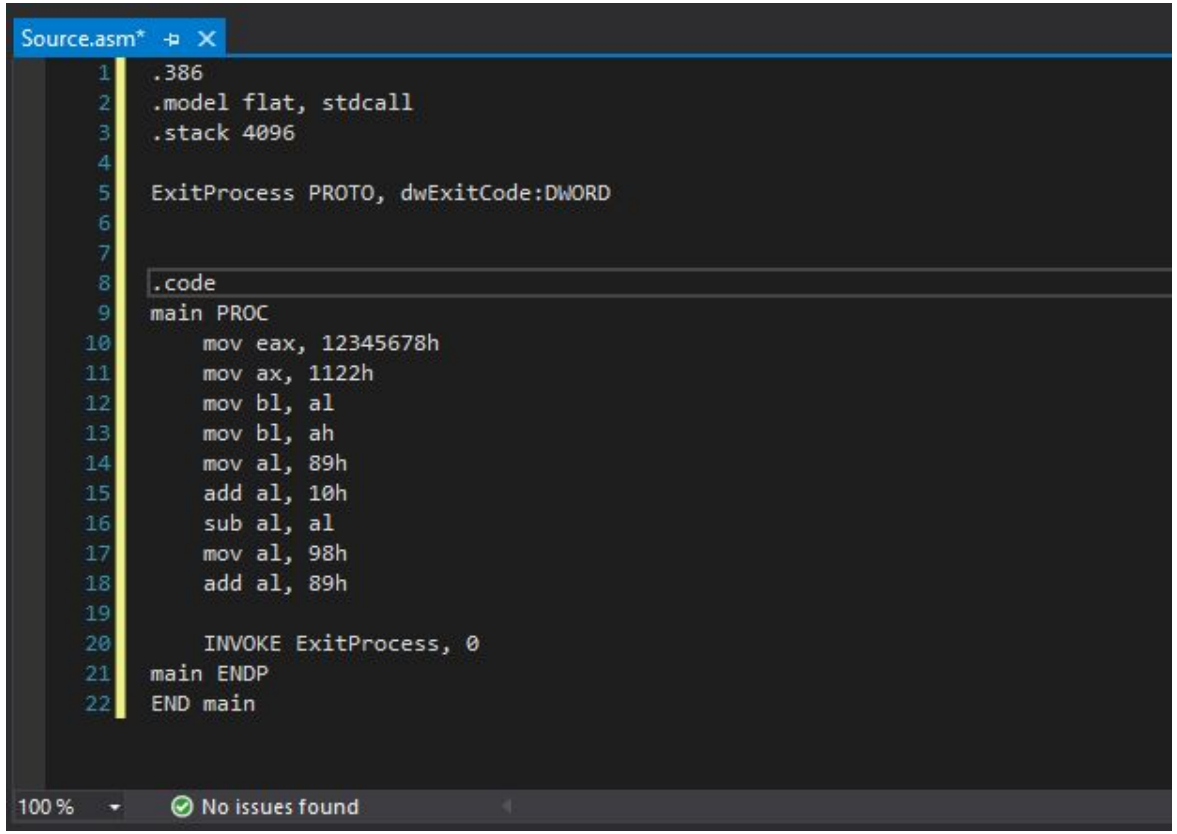
- Once the project has been opened, you will see the project name in the Solution Explorer window.
  - You should also see an **assembly language source file** in the project named **Source.asm**.
  - **Double-click** the file name to open it in the editor.



Zulkar Nine (email: mnine@gsu.edu)

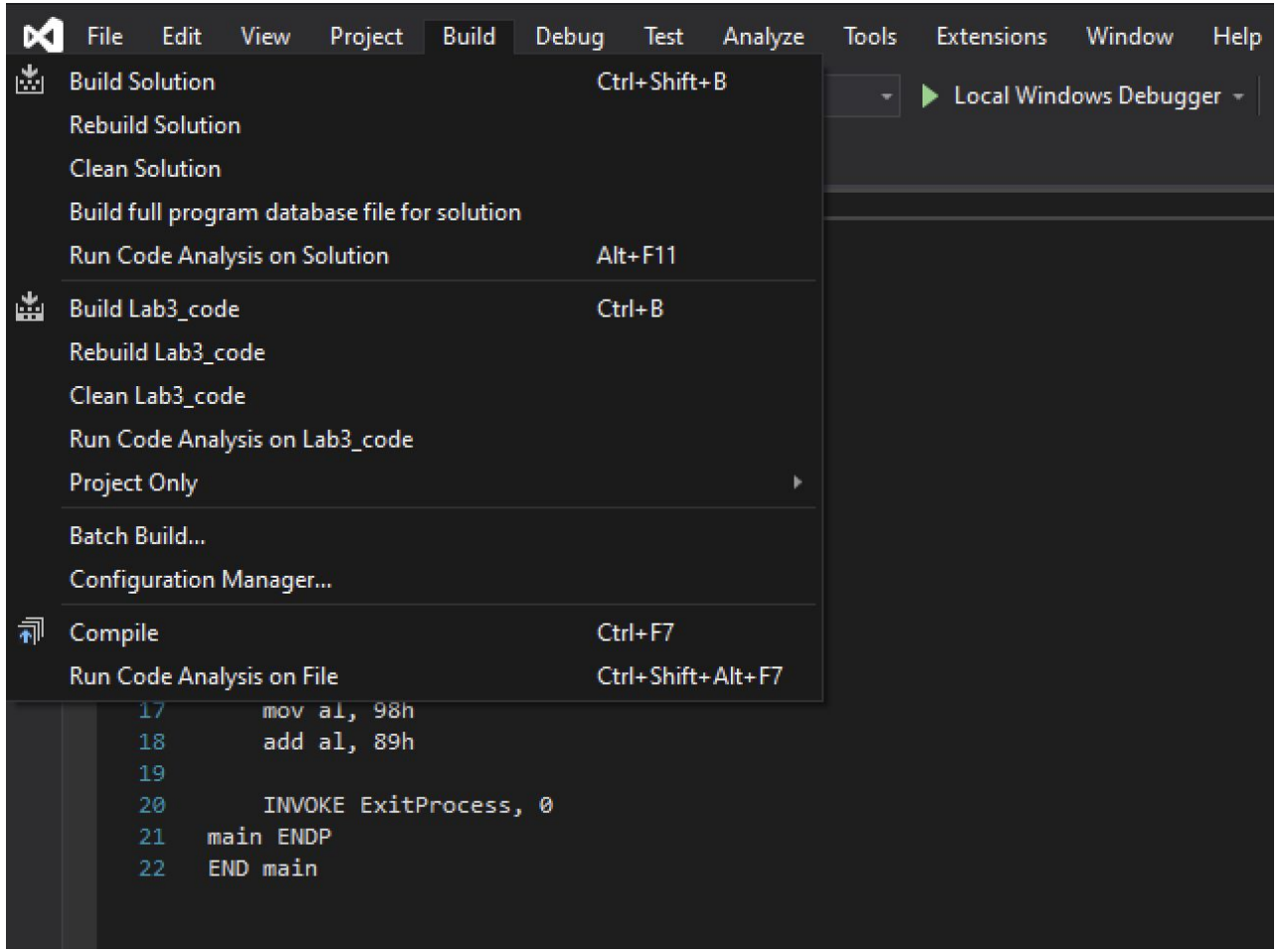
Step 2: Load the project into Visual Studio (4)

- You should see the **source code in the editor window**



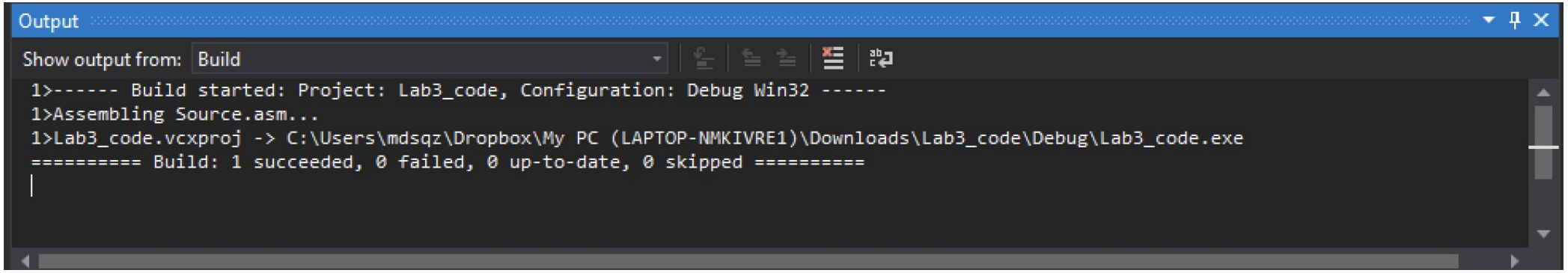
Step 3: Build the Project (1)

- Select **Build Project** from the **Build menu**.
- This will **assemble** and **link** your program and create an executable file.



Step 3: Build the Project (2)

- You should see messages like the following in your output window, indicating the build progress:



- You should see the message in the last line:

=====Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====

Lab 3(b)
Debug the project code

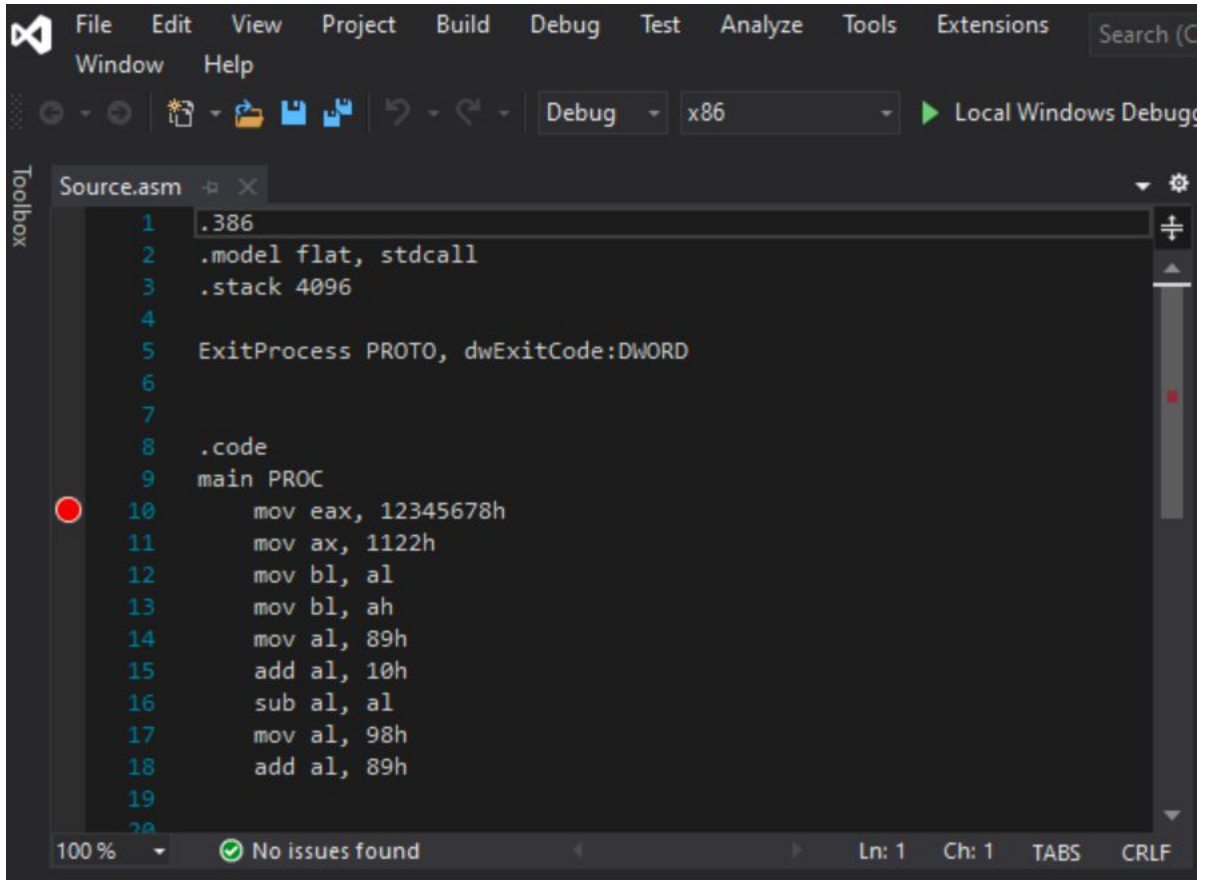
And record the register and flag contents

Lab 3(b) Submission Instructions

- An answer sheet is provided with this lab for Lab 3(b).
- You need to debug the code
- Stop for each instruction and record the register content in the answer sheet. Then explain the register content.
- Submit your answer sheet to the iCollege.

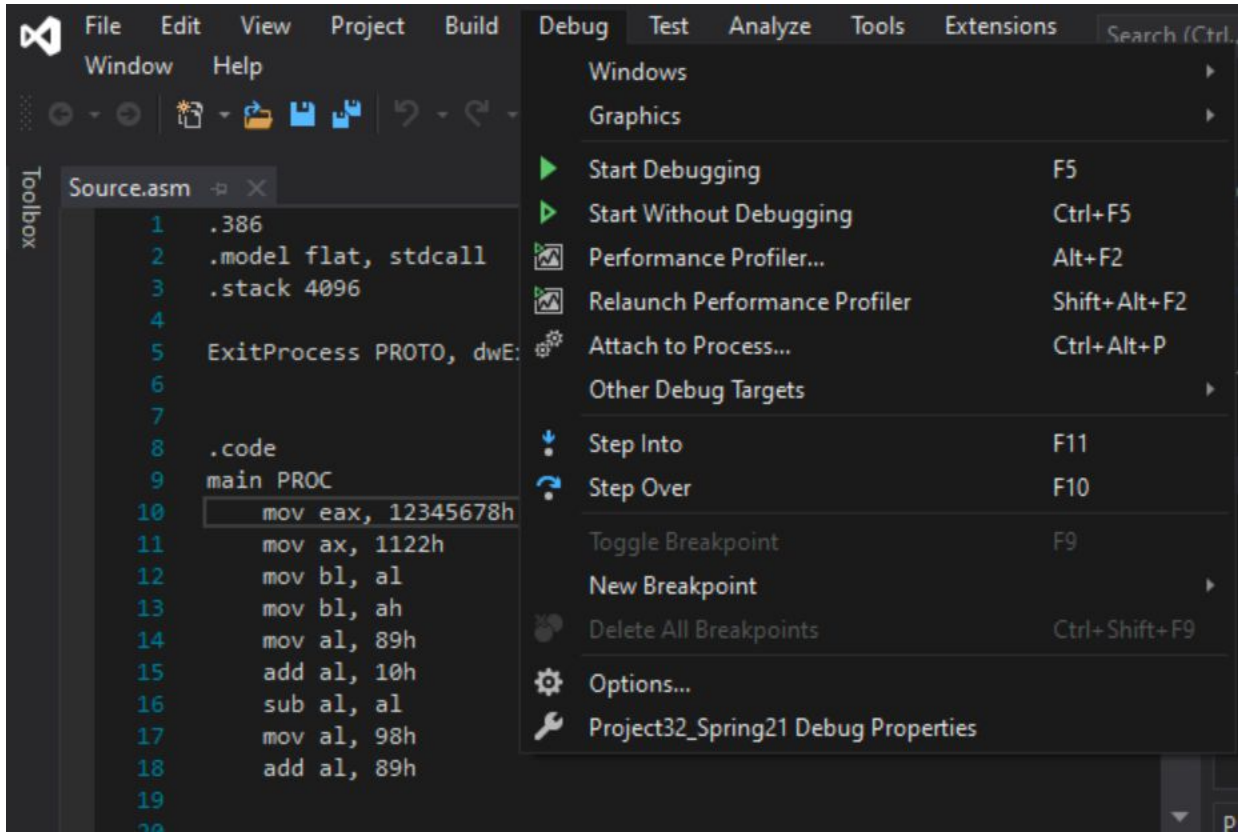
Debug the Project

- Set a breakpoint first.
- **Set a breakpoint** on a program statement by clicking the mouse in **the vertical gray bar** just to the left of the code window.
- A **large red dot** will mark the breakpoint location.
- In this case, set a breakpoint at Line 10.



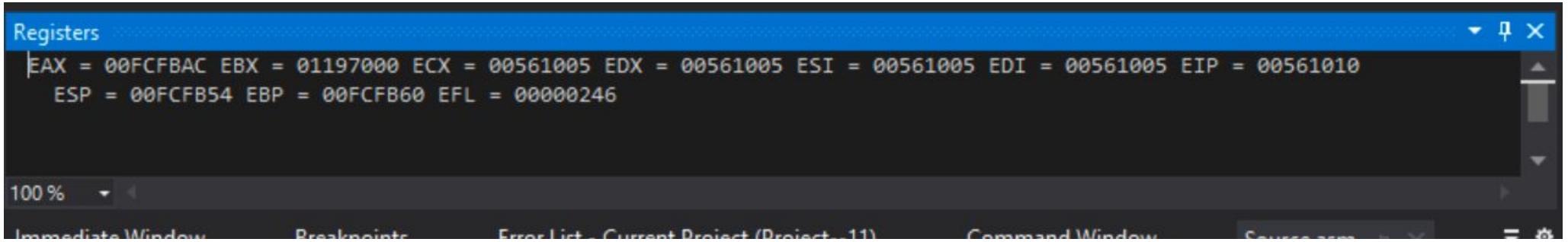
Debug your code

- **Run** the Program by selecting **Start Debugging** from the **Debug menu**.



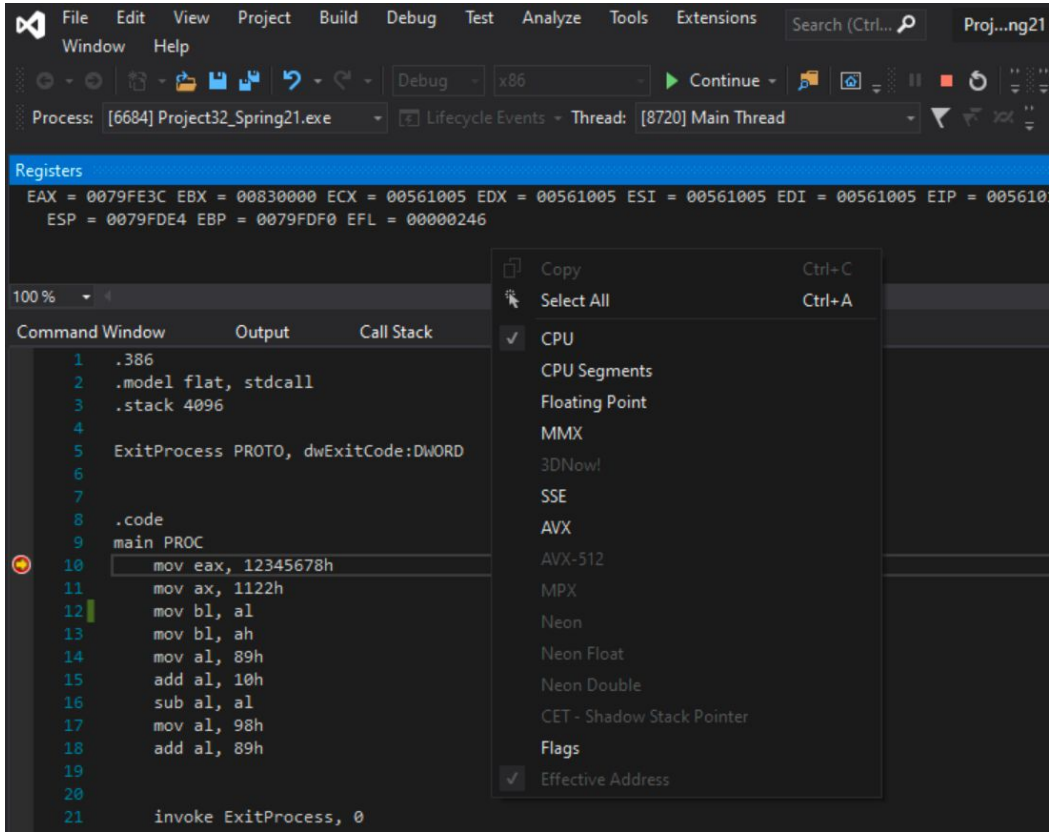
Debug the Project

- You should be able to see the register window :



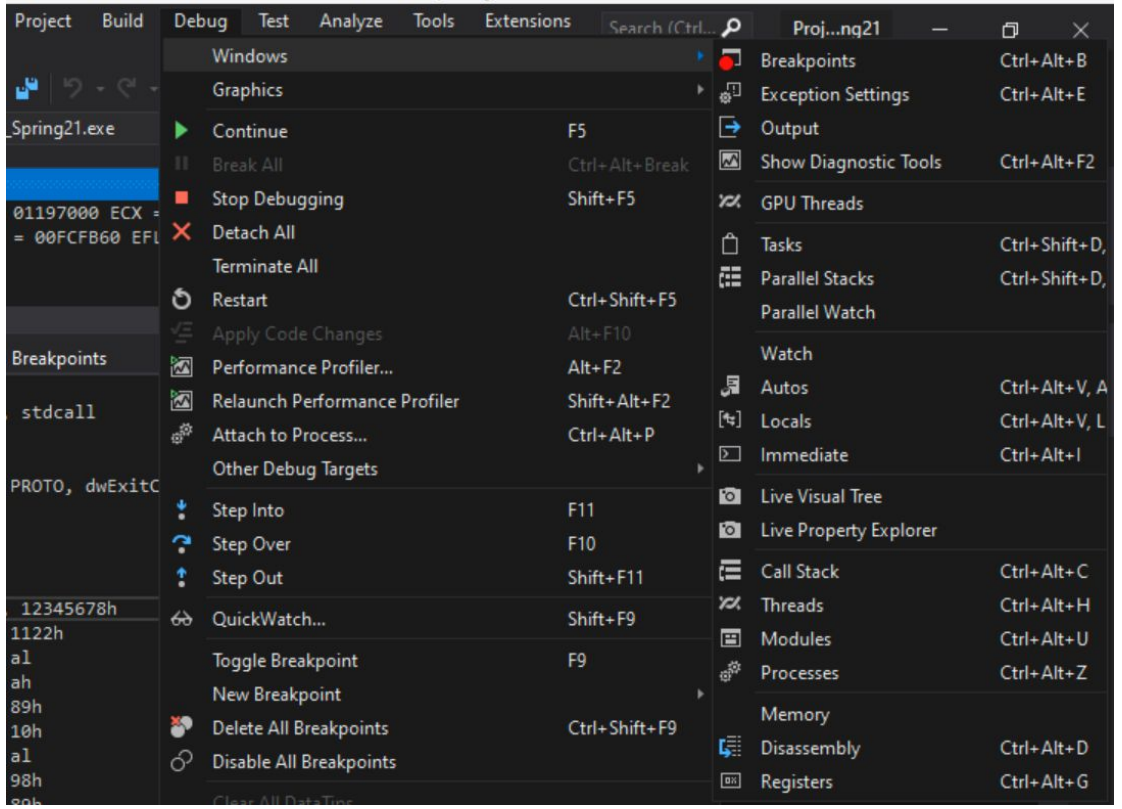
Debug the project

- Turn on the EFLAGS in the register window.
- Right click on the register window
- Then check the flag



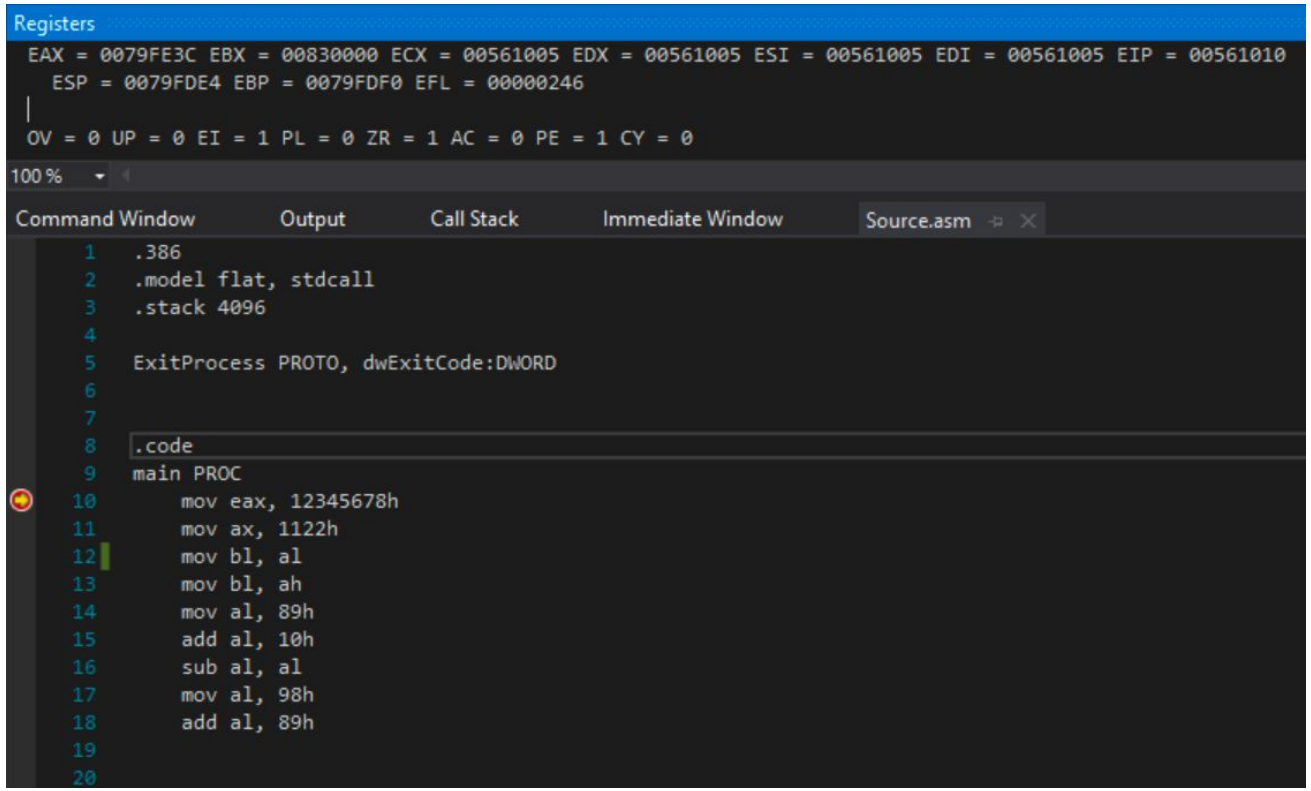
Debug the Project

- You must be inside debug mode:
  - If you are not in debug mode : Go to Debug -> Start debugging
- If you don't see the register window: Go to : Debug ->Windows->registers.



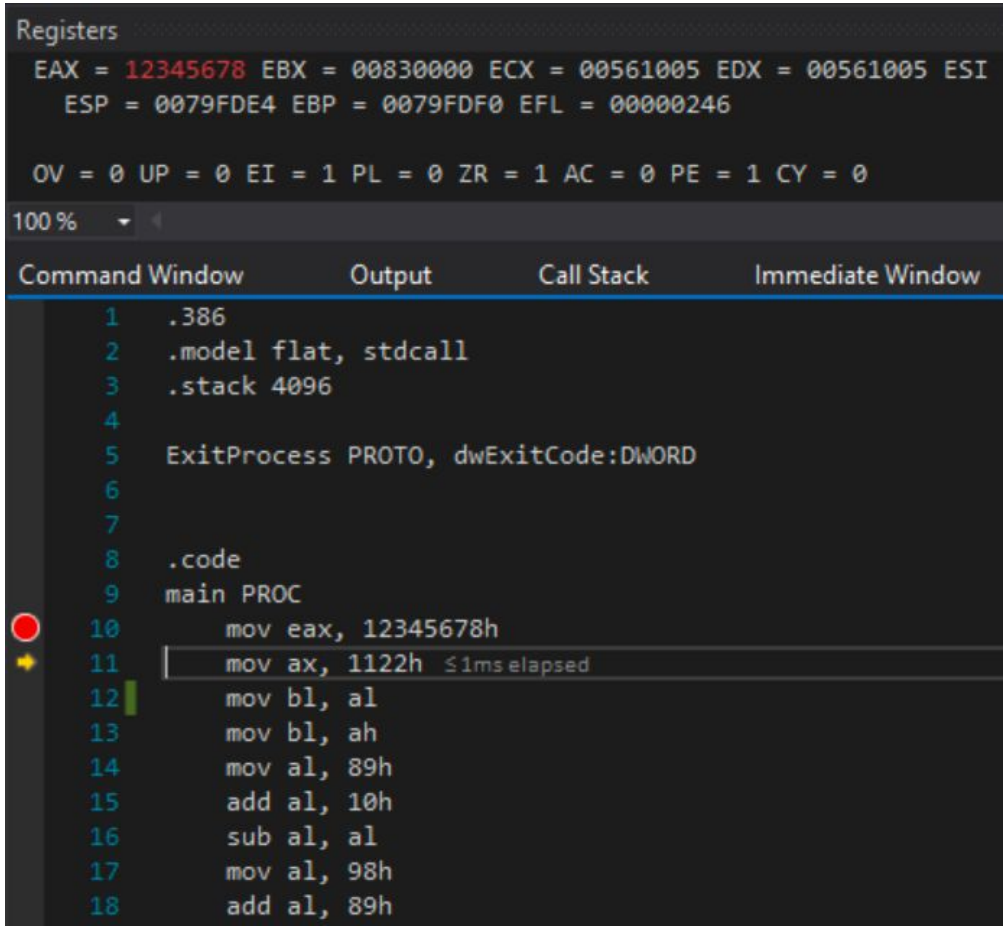
Debug your code

- Now the red dot (breakpoint) has a yellow pointer inside of it now.
- That means code execution halts at line 10 now.



Debug the Project

- To execute line 10, Select 'Step over' from debug menu
  - You can also use shortcut : Fn+F10
- Now the yellow pointer moved to line 11. That means line 10 is executed.
- Check the EAX register content.

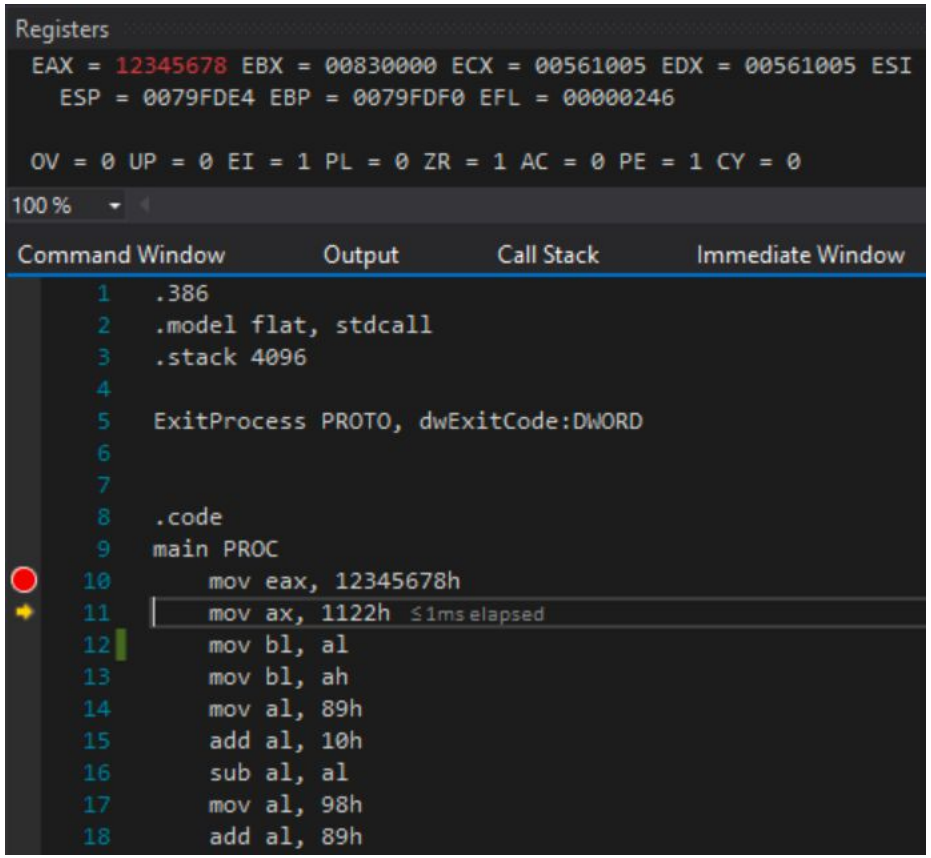


mov Instruction

- mov instruction has two inputs – first one is destination and second one is source.
  - mov destination, source
  - mov instruction copy the source content to the destination.
  - It is more like assignment operation in high level language.
- In the Code, Line 10
  - Mov eax, 12345678h
  - Here 12345678 is a hexadecimal number. Radix 'h' after the number indicates that it is a hex.
  - This can be converted to 32-bit binary number. (Try convert it to binary)
  - Mov instruction stores that value to EAX register that is also 32-bit register.
- After executing line 10, you can see the content change in EAX register.

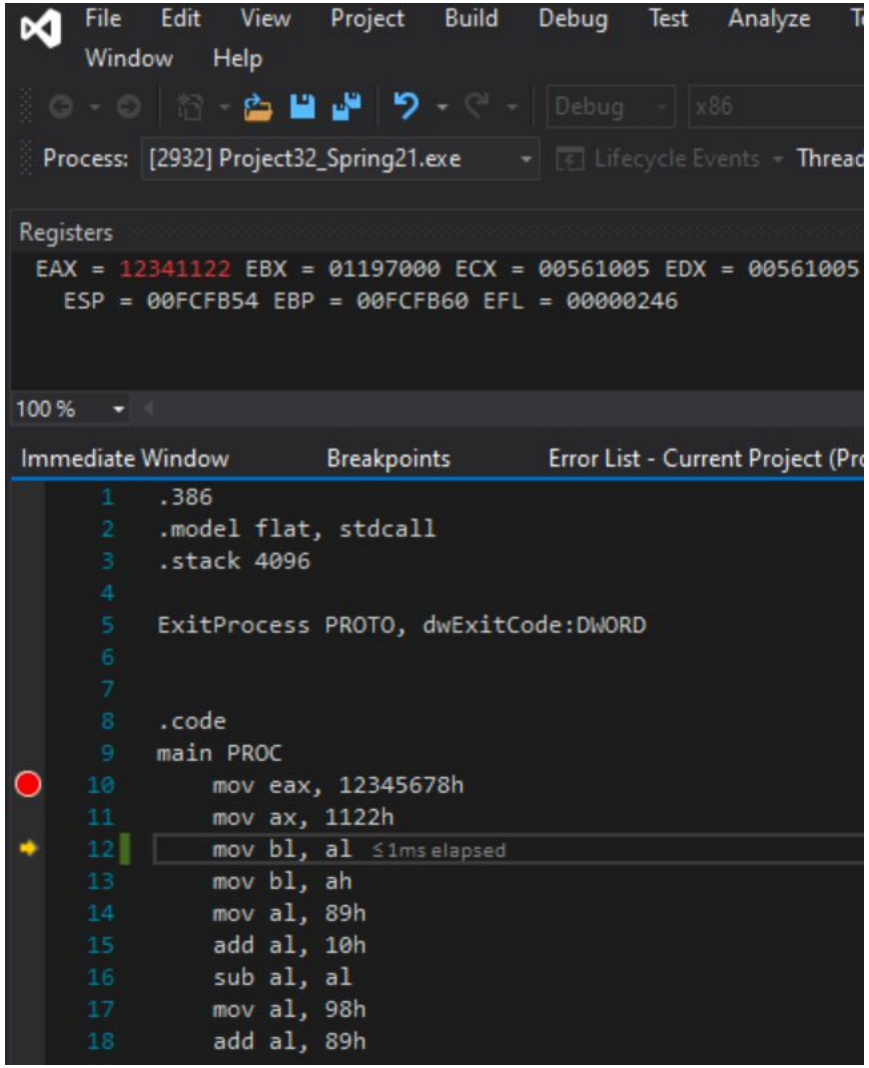
mov Instruction

Eax contains 12345678 (all the register values are in hexadecimal)

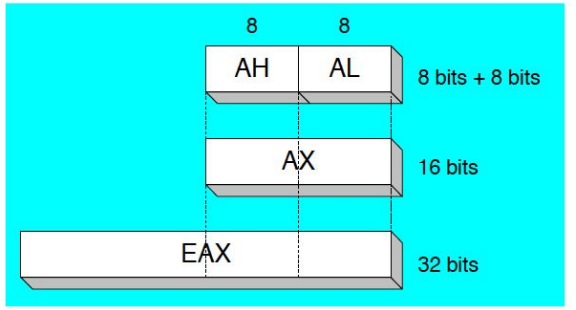


Debug : Line 11

- To execute line 11, Select ‘Step over’ from debug menu
  - You can also use shortcut : Fn+F10
- Mov ax, 1122h
  - Here 1122 is a hex and it is 16 bit.
  - AX is also a 16-bit register.
- After executing the mov instruction,
  - Only AX part (lower) of EAX register updated.
  - The upper part is unchanged.

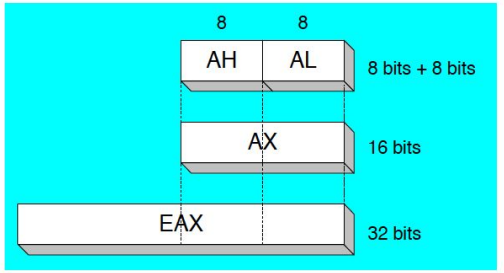


Debug: Line 12

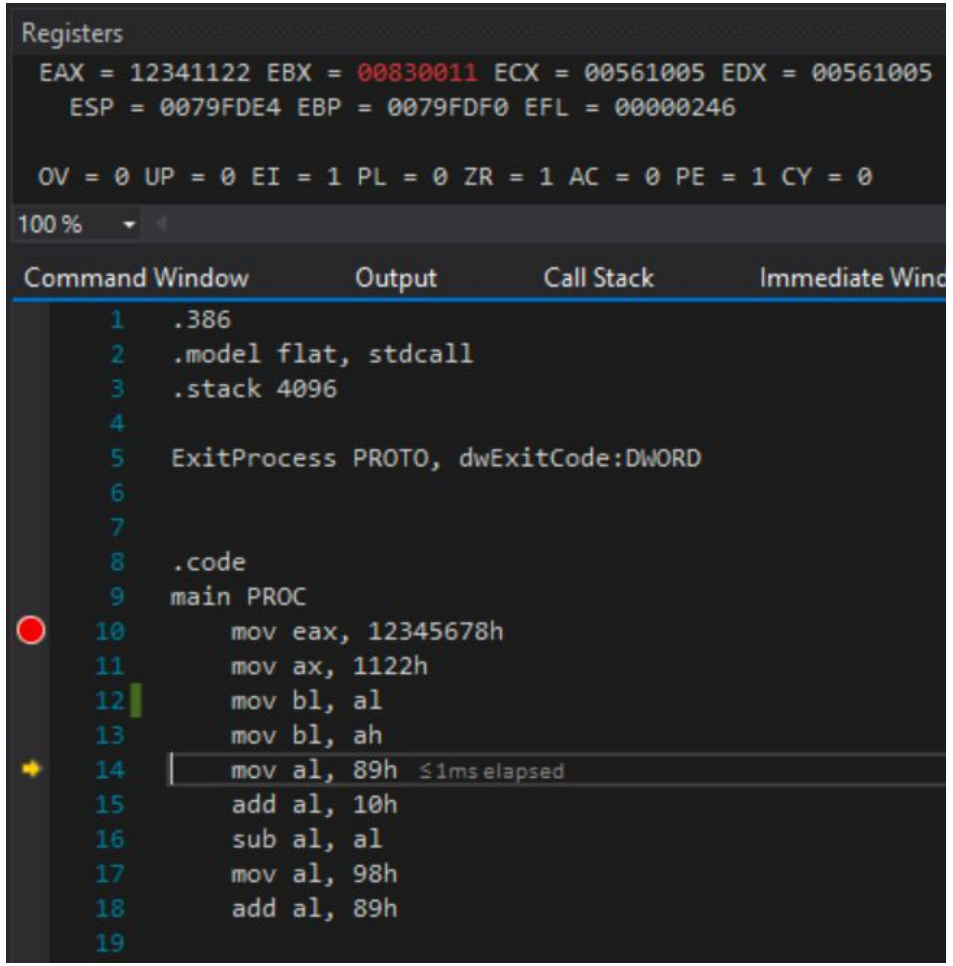


- To execute line 12, Select 'Step over' from debug menu
  - You can also use shortcut : Fn+F10
- Mov bl, al
  - Here the content of AL register is moved to the BL register.
  - AL register is the lower 8-bit of the EAX register.
  - It contains 22 now.
  - 22 is a hex
  - BL is the lower 8 bit of EBX register
  - BL is the destination here
- After executing this line, BL contains 22 as well. However, rest of the EBX register remains unchanged with previous garbage values.

Debug: Line 13



- To execute line 13, Select 'Step over' from debug menu
  - You can also use shortcut : Fn+F10
- Mov bl, ah
  - Here the content of AH register is moved to the BL register.
  - AH register is the upper 8-bit of the AX register.
  - It contains 11 now.
  - 11 is a hex
  - BL is the lower 8 bit of EBX register
  - BL is the destination here
- After executing this line, BL contains 11 as well. However, rest of the EBX register remains unchanged with previous garbage values.



Add and Sub instruction

- add instruction has two inputs – first one is destination and second one is source.
  - add destination, source
  - add instruction adds the source content to the destination and store the result in destination.
- For example,
  - add al, 12h
  - Adds 12 to the content of AL register and store the result in AL register
  - It is more like : al = al + 12 in a high level language
- sub instruction is similar to add instruction
  - Instead of addition it performs subtraction.

Submission : Lab 3(b)

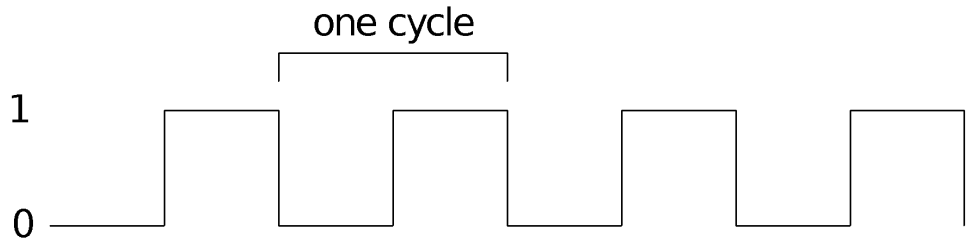
- Debug the rest of the code one line at a time.
- There is an answer sheet provided with the lab.
- After executing each line record the register value in the answer sheet.
- Then also provide short explanation for the changes.
- Submit the filled out answer sheet at iCollege.

Processor Clock

A review

Clock

- Each processor has a built in clock to synchronize the internal operations.
- Each CPU operation synchronized by an internal clock pulsing at constant rate.



- A **clock** is a sequence of **1's** and **0's**
- **Clock Cycle: a 0 and 1 produce a clock cycle**
- **Frequency: The number of cycles happens per second**
- Unit : Hz = 1 cycle per second

Clock

- Clock period = time length of a clock cycle

$$\text{Clock period} = \frac{1}{\text{Clock frequency}}$$

- A CPU has a clock frequency 1 GHz. What is the clock period?
  - Clock period = $\frac{1}{10^9} = 1\ nenosecond$
- Let's say an instruction, takes 40 clock cycle to execute in your 1 GHz processor. What is the actual time it takes to execute the instruction?
  - Clock period = 1 ns, so each cycle takes 1 ns to finish
  - 40 clock cycle takes = 40 * 1 ns = 40 ns to finish
- So the instruction takes 40 ns to execute.

An example

- Suppose a program contains 1 billion instructions to execute on a processor running on 2 GHz. The instructions takes 3 clock cycles to execute. What is the execution time of the program?
- Answer:
  - 1 billion instructions each takes 3 clock cycles
  - Total clock cycle for the program = 1 billion * 3 cycles = $3 * 10^9$cycles.
  - Given, Processor Frequency = 2 GHz
  - Processor produces $2 * 10^9$ cycles in 1 second.
  - $2 * 10^9$ cycles takes 1 second
  - $3 * 10^9$ cycles takes $\frac{3*10^9}{2*10^9}$ second = 1.5 seconds.

Clock per Instruction (CPI)

- Is an effective average.
- It is the average number of clocks required by the instructions in a program.
- In a program 30% instructions takes 4 clock cycles and the rest of the instructions takes 1 clock cycles.
- CPI = 0.3 * 4 + 0.7 * 1 = 1.9 clocks per instruction.

Million Instructions Per Second

- **Step 1:** Perform Divide operation between no. of instructions and Execution time.
- **Step 2:** Perform Divide operation between that variable and 1 million for finding millions of instructions per second.
- For example,
  - if a computer completed 2 million instructions in 0.10 seconds
  - 2 million/0.10 = 20 million.
  - No of MIPS=20 million/1 million
  - =20

Lab 3(c)
Some math Problems

Submit the problems

Lab 3(c) : Submission

- Solve the Problems provided in slide 45 and 46.
- You can do your work in a text editor (Microsoft word, open office, etc.)
- Or you can do it in a piece of paper, then scan or take a picture of the paper.
- Convert them into pdf and submit in the iCollege.

Problem 1

- Suppose a program contains 500 million instructions to execute on a processor running on 2.2 GHz. Half of the instructions takes 3 clock cycles to execute, where rest of the instructions take 10 clock cycle. What is the execution time of the program?

Problem 2

- A processor is 20 MIPS. If you run a program on that processor and the program takes 30 seconds to finish. How many instructions are there in this program?