Digital Design & Computer Arch.

Lab 9 Supplement: The Performance of MIPS

(Presentation by Aaron Zeller)
Frank K. Gürkaynak
Seyyedmohammad Sadrosadati
ETH Zurich

Spring 2024

[28. May 2024]

What we have done so far

- You have learned how gates work and how to build small modules out of them.

- You combined those modules into more powerful ones.

- You learned how to test code via simulation and test benches.

- You learned how to write MIPS assembly code.

- And finally: You combined all of that knowledge to build a simple processor and even ran your own programs on it.

Lab 9 Overview

- Extend the processor from the previous lab: srl, mflo, multu
- This requires you to make the aluop signal larger to support the additional operation.
- Measure the performance of the processor.

Lab 9: New Instructions

- The processor can already add and subtract but to be able to efficiently compute we want to add multiplication.

- By shifting we can multiply and divide by powers of 2.

- You are only required to implement a logic right shift (srl)
  - but feel free to implement an arithmetic right shift and logic left shift.

- Every left shift can be implemented by a normal multiplication, so we do not need it.

Lab 9: How to do 32-Bit Multiplication

- The result of a multiplication of two 32-bit numbers can have up to 64 bits.

- Two stage approach:
  - MIPS has special registers (Hi and Lo) where the upper 32 bits of the multiplication result are saved in Hi and the lower 32 ones in Lo (no output from the ALU.)
  - mfhi and mflo can then retrieve the values.

Lab 9: Performance

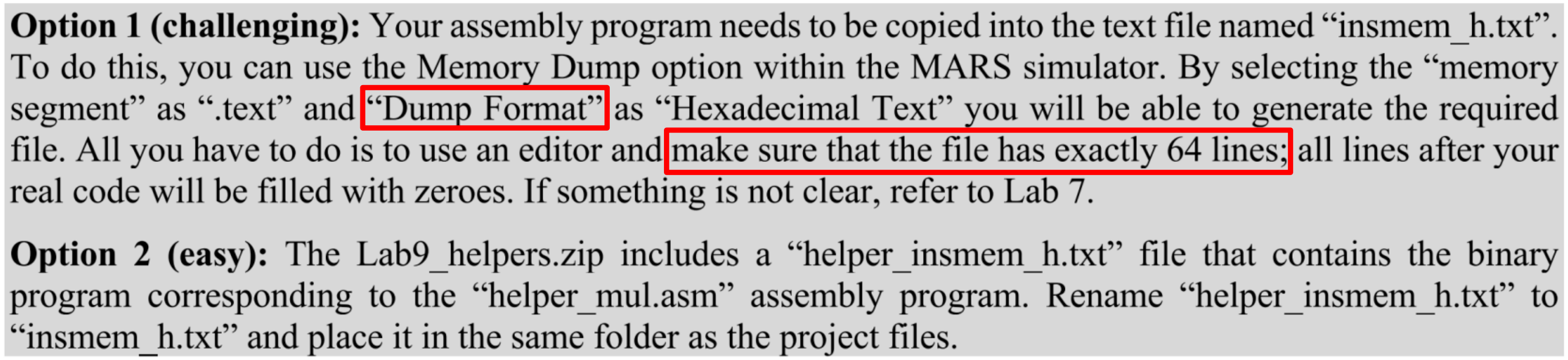
- We want to determine the performance of our processor on the example that you implemented in Lab 7: summing up numbers:
  - Once with a loop that performs back-to-back additions.
  - Once with the Gaussian formula that performs multiplication.

- You can use your implementation from Lab 7 and write a new code for calculating the Gaussian formula or use the code that we provide.

- Finally, you will test your processor with a test bench.

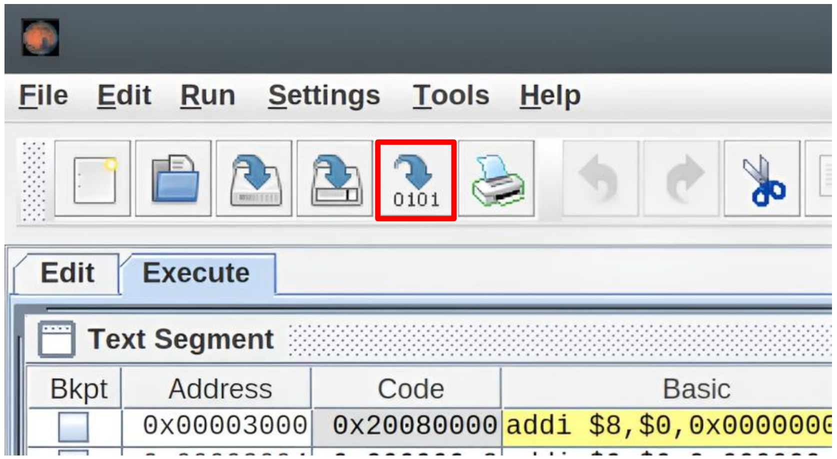
Lab 9: Verification

- The MIPS assembly file you produce the hexadecimal text from must be exactly 64 lines. This includes empty lines.



Option 1 (challenging): Your assembly program needs to be copied into the text file named "insmem_h.txt". To do this, you can use the Memory Dump option within the MARS simulator. By selecting the "memory segment" as ".text" and "Dump Format" as "Hexadecimal Text" you will be able to generate the required file. All you have to do is to use an editor and make sure that the file has exactly 64 lines; all lines after your real code will be filled with zeroes. If something is not clear, refer to Lab 7.

Option 2 (easy): The Lab9_helpers.zip includes a "helper_insmem_h.txt" file that contains the binary program corresponding to the "helper_mul.asm" assembly program. Rename "helper_insmem_h.txt" to "insmem_h.txt" and place it in the same folder as the project files.



Bkpt	Address	Code	Basic
☐	0x00003000	0x20080000	addi $8,$0,0x0000000

Last Words

- In this lab, you will add instructions to the processor.
  - Adding instructions improves the processor design.
  - But this comes at a cost: more instructions have to be decoded and more hardware resources are required.

- Processor designers need to face these **trade-offs**.

- We hope you have enjoyed the labs.
  - You have implemented your own 32-bit processor.
  - You have written programs for it.
  - You have improved its performance.

Report Deadline

[14. June 2024 23:59]

Digital Design & Computer Arch.

Lab 9 Supplement: The Performance of MIPS

Frank K. Gürkaynak
Seyyedmohammad Sadrosadati
ETH Zurich

Spring 2024

[28. May 2024]