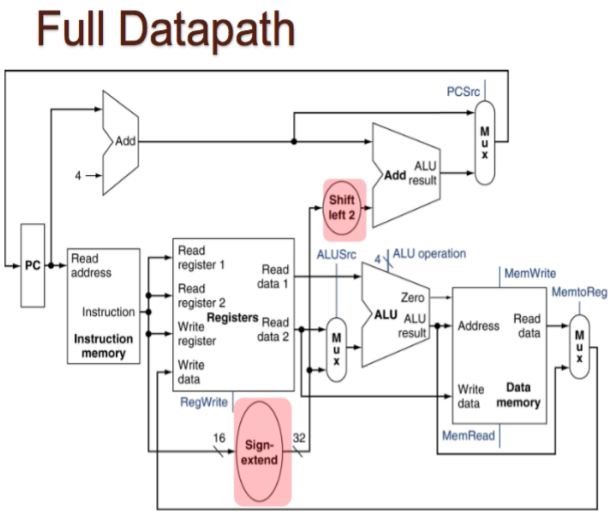
10420 CS410001 - Computer Architecture 2016

Project #1

1. Project Objective
   a. Implement a single-cycle functional processor simulator.
   b. Design your own test case to test the functionality of your simulator.

2. Project Description
   a. Architecture Design:
      - Refer to textbook *Chapter 2 "Instructions"* and *Chapter 4.1~4.4 "The Processor"*.
      - The executable file should be named "single_cycle" and take **no command-line arguments.**
      - All registers, except PC and $sp, are initialized to 0's.
      - Assume that the instruction memory is of 1K bytes size and all contents are initialized to 0's.



      - The data memory is of 1K bytes size and the memory contents are initialized to 0's.
      - Your test case should run no more than 500,000 cycles.

   b. Instruction Set:
      - Implement all the instructions specified in the reduced MIPS R3000 ISA in ***Appendix A*, "*Datasheet for the Reduced MIPS R3000 ISA*"**.
      - The execution of the single-cycle processor simulator should **terminate** after executing the **"halt" instruction**.

   c. Input Test Case File and Format:
      - Design your own test case by providing the following two **binary files**:
        1. **iimage.bin**:
           a. This file specifies the instruction image (in **big-endian** format, encoded in binary).
           b. The first four bytes indicate the initial value of PC, i.e. the starting address to load the instruction image.
           c. The next four bytes specify the number of words to be loaded into instruction memory.
           d. The remaining are the program instructions to be loaded into I-memory.

2. **dimage.bin**:
    a. This file specifies the data image (in **big-endian** format, encoded in binary).
    b. The first four bytes indicate the initial value of $sp.
    c. The next four bytes specify the number of words to be loaded into data memory, starting from address 0.

- For format details, please refer to ***Appendix B*, "*Input Samples*"**.
- Place both "iimage.bin" and "dimage.bin" under a folder named testcase/ at the same directory where your executable file resides.
- We will pool all test cases from the class to evaluate everyone's simulator. Your (valid) test case gets higher grade if more simulators failed running your test case.

d. Output File and Format:
- For each test case, generate the following two output files:
    1. **snapshot.rpt**: Record all the register values at each cycle.
    2. **error_dump.rpt**: Record any error messages.
- Place the output files at the same directory where your executable file resides.
- For details, please refer to ***Appendix C-1, "Output Samples for Project 1"*** and ***Appendix D, "Error Detection Samples"***.

e. Modularized implementation
- Suggest that you should **modularize** your simulator implementation based on processor architecture. For example, this is a possible program structure:
    a. simulator.c // Define simulator behaviors and main function
    b. instruction.c // Define & decode instructions
    c. regfile.c // Register function
    d. memory.c // Memory function (for both instruction & data memories)
    e. etc.c…… // other miscellaneous functions
- Appropriate header file or object-oriented programming format are also highly recommended design pattern.

3. The 2-submission Process
a. Each project has two submissions. The second due date is normally one week after the first one.
- Before 1st submission: we will release a **golden executable** and **open test cases** to help you verify your designs.
- 1st submission: submit your simulator and test case for evaluation.
- After 1st submission (before 2nd submission): we will release **all test cases (including hidden test cases and all test cases submitted by the class)** for you to polish your

simulator.

- 2nd submission: submit your revised simulator and project **report** (format is specified in the Grading Policy section).

b. Prepare your project package for development and submission

- Before you start your project
  1. Use SSH to access workstation.
  2. Clone sample files from GitHub to your home directory.
     - Clone a repository NTHU Architecture 2016 from GitHub and name it as studentID_01/ under /home/archi/studentID. Inside the folder, it contains two folders:
       i. simulator/ : contains your Makefile, and source code.
          a. Your Makefile should support the following two functions:
             - make – to build your simulation environment
             - make clean – to erase from the build tree the files built by make all.
          b. Modulize your source code as recommended.
       ii. testcase/ : contains your test case files for evaluation.
  3. Do coding/debugging/testing using Git version control tool.
- Before submission and after completing your project
  1. Check your output file format using test_script.py
  2. Compress the folder studentID_01 as studentID_01.tar.gz, and upload studentID_01.tar.gz and studentID_report.pdf to the iLMS system.

- **Note:** TAs will check your Git log during 1-on-1 Demo. Please follow version control rule when doing programming.
- **Note:**. Verify your package format by executing test_script.py before each submission. **Wrong submission format earns no points.**

4. Grading Policy

a. First submission

- Correctness of simulator: 25%
  1. TA's open test cases: 15%
  2. TA's hidden test cases: 5%*Correct Ratio
  3. Students' valid test cases: 5%*Correct Ratio
- Test case strength: $20\%*(1 - [1.5]^{-n})$, n: number of other simulators defeated
  1. Submit your test case to participate in the pool test.
     - **Note:** You get zero points if your test case is invalid.

b. Second submission
   - Correctness of simulator: 30%
     1. TA's open test cases: 5%
     2. TA's hidden test cases: 10%*Correct Ratio
     3. Students' valid test cases: 15%*Correct Ratio
   - Performance: 5%
     1. TA will collect the execution time of your simulator running all the valid test cases (including all open, hidden and students' test cases).
     2. TA will rank all execution times in 5 levels and grade accordingly.
     - **Note:** If your simulator fails to execute the valid test cases you will get the lowest performance grade.
   - Report & Demo: 20%
     1. The report file should be named ***studentID_report.pdf***, where ***studentID*** is the NTHU student id you used for school registration.
     2. Each person should reserve a 15-min demo with TA. During the 1-on-1 demo, TA's will ask you questions related to your project report, test case and your implemented code on workstation.
     3. Your project report is recommended to follow this outline:

1) Project Description
1-1) Program Flow Chart
1-2) Detailed Description
2) Test case Design
2-1) Detailed Description of Test case

Note: The project report is limited to 10 pages.
Note: Your report can be either in Chinese or English, or mixed.
Note: For convenience, please synchronize your submission code with your code in workstation. This allows smoother demo on workstation.

- **Etiquette**
  a. **Do not plagiarize others' work, or you will fail this course.**
  b. **No acceptance of late homework.**
  c. **Please frequently check the class website announcements for possible updates.**