1052 Computer Architechture Final Project

Pipelined MIPS Design

Announced at May 18, 2017

TA Information

周敬堯，電二 232 實驗室

endpj@access.ee.ntu.edu.tw

1. Project Description

Table 1. Required Instruction Set

Name	Description
ADD	Addition, overflow detection for signed operand is not required*
ADDI	Addition immediate with sign-extension, without overflow detection*
SUB	Subtract, overflow detection for signed operand is not required*
AND	Boolean logic operation
ANDI	Boolean logic operation, zero-extension for upper 16bit of immediate
OR	Boolean logic operation
ORI	Boolean logic operation, zero-extension for upper 16bit of immediate
XOR	Boolean logic operation
XORI	Boolean logic operation, zero-extension for upper 16bit of immediate
NOR	Boolean logic operation
SLL	Shift left logical (zero padding)
SRA	Shift right arithmetic (sign-digit padding)
SRL	Shift right logical (zero padding)
SLT	Set less than, comparison instruction
SLTI	Set less than variable, comparison instruction
BEQ	Branch on equal, conditional branch instruction
J	Unconditionally jump
JAL	Unconditionally jump and link (Save next PC in $r31)
JR	Unconditionally jump to the instruction whose address is in $rs
JALR	Jump and link register
LW	Load word from data memory (assign word-aligned)
SW	Store word to data memory (assign word-aligned)
NOP	No operation

In final project, you are asked to design a **pipelined MIPS processor with instruction cache and data cache**. This processor should at least support the instruction set defined in Table 1. The instruction set is referenced from Appendix A of [1], and we encourage you read it in detail.

* Different from definition in [1], the exception handler for arithmetic overflow is not required.

The processor architecture in Figure 1 is referenced from Chapter 4 of [1]. As you see, this is modified from single-cycle architecture of our HW3. Your design should follow this 5-stage pipelined architecture. You need to modify several parts to fit our specifications. For example, you need to add the path for **J-type instructions**.

Also, you should **solve the hazards** by adding some circuits. There are 3 hazard categories should be properly handled in your pipelined processor:

1) Structure hazard;

2) Data hazard;

3) Branch hazard.

Although all of these hazards can be solved by insert NOP manually or automatically in your test program, we ask you to implement **data forwarding unit** and **pipeline stall unit** to solve these hazards.

2. Cache and Memory Interface

The instruction memory and data memory will not be contained in your design. The memory interface is left as module I/O. You have to use the provided slow memory model but don't have to synthesize them.

The cache units is suggested to have the same block number (8) and block size (4) as in HW5. Besides, we do not restrict the replacement policy and writing policy of the cache design. You are encouraged to optimize the cache units to fit your MIPS design.

3. Synthesis Notes

You should synthesize your design using TSMC 0.13 cell library, and the relevant files, e.g. `.synopsys_dc.setup`, can be copied from previous HW and Labs. The design constraints is specified in "CHIP_syn.sdc".

Note that the pipelined MIPS, instruction cache and data cache are included in the CHIP.v and they should be synthesized together.

The post-synthesis simulation is required and all involved Verilog files should be all modeled by gate-level. Note that the maximum clock frequency must be verified by post-synthesis gate-level simulation.

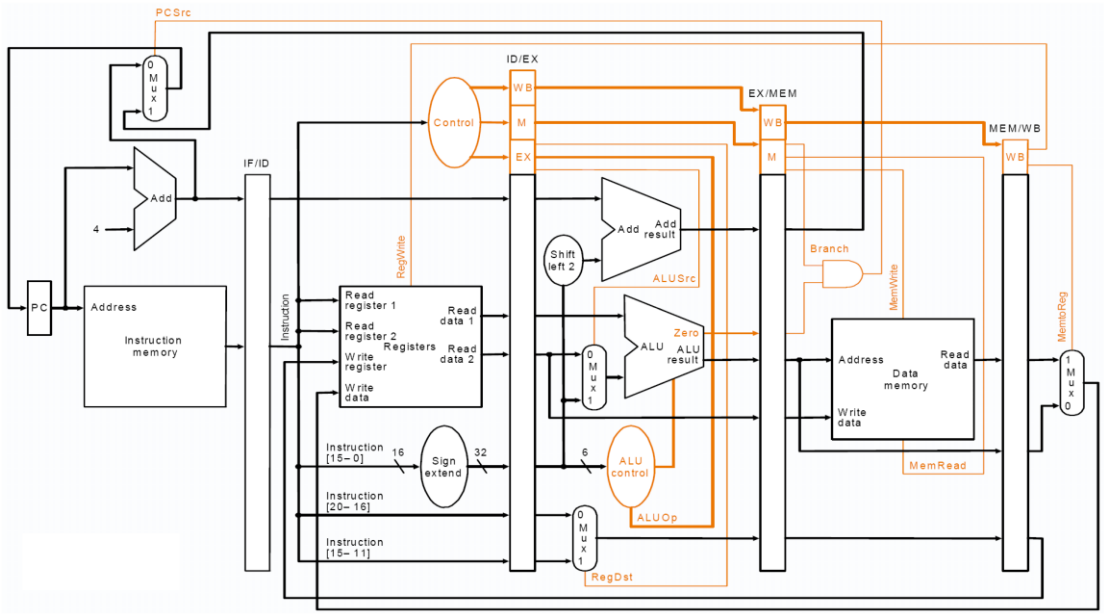

Figure 1. Simplified Pipeline Architecture of MIPS.

4. Grading Policy and Possible Extensions

All grades of this project consist of three equal aspects:

1) Technical features; 2) Presentation; 3) Final report.

The technical features are grading by two parts:

1) Baseline; 2) Extension. The details are as follows.

Baseline (50%)

If your design meets all requirements of above description, and can be synthesized and simulated at gate-level, your design will get baseline points. The solid requirements include:

1) Supporting all instructions above
2) With caches
3) Pass all test assembly programs
4) Complete the circuit synthesis

Then the performance is evaluated by (the smaller the better):

Area (um^2) * Total simulation time (ns);

Extension (50%)

There are four topics of extension.

1) Branch prediction mechanism.
2) Two-level caches, i.e. with L2 caches
3) Implement assembly code written in hw2
4) Supporting multiplication and division.

Implement **as much as you can** of the topics of extension

5. Simulation example

RTL level

ncverilog Final_tb.v CHIP.v slow_memory.v **+define+noHazrd** +access+r

Gate Level

ncverilog Final_tb.v CHIP_syn.v slow_memory.v tsmc13.v
+define+noHazrd +define+SDF +access+r

6. Schedule and Necessary Submissions

Date	Submission/Event
5/25	Announcement of Extension details
6/01	A. Checkpoint. Each team should prepare a proposal (*1-2 pages word and 4-6 pages powerpoint (about 5 minutes)*) to confirm your current results and future plan. You should upload the team proposal to the FTP by the day. B. Extension topics plan should be included in the proposal
6/15	Final presentation. Each team should prepare a full talk (*within 15 minutes*, about 10-20 slides) to demonstrate your fantastic work! Detail presentation plan will be announced in the ceiba.
6/23	Final submission, including a detailed report (*8-16 pages*), the presentation slides, and all the source codes (including all the RTL code and synthetis related files: *.vg, *.sdf, *.ddc and a Readme.txt). You should upload the final submission to the FTP by the day.

7. Reference

[1] David A. Patterson and John L. Hennessy, *Computer Organization & Design: The Hardware/Software Interface*. Morgan Kaufmann, 1998