Computer Architecture – Project #1

MIPS Single Cycle CPU Implementation

1. Introduction

We look at what might be thought of as the simplest possible implementation of our MIPS subset. This simple implementation covers load word(lw), store word(sw), or immediate (ori), and jump(j). Constant manipulation instruction (lui) and user defined instructions (llo, lhi) are also implemented. The following figure shows an example of the similar single-cycle CPU implementations.

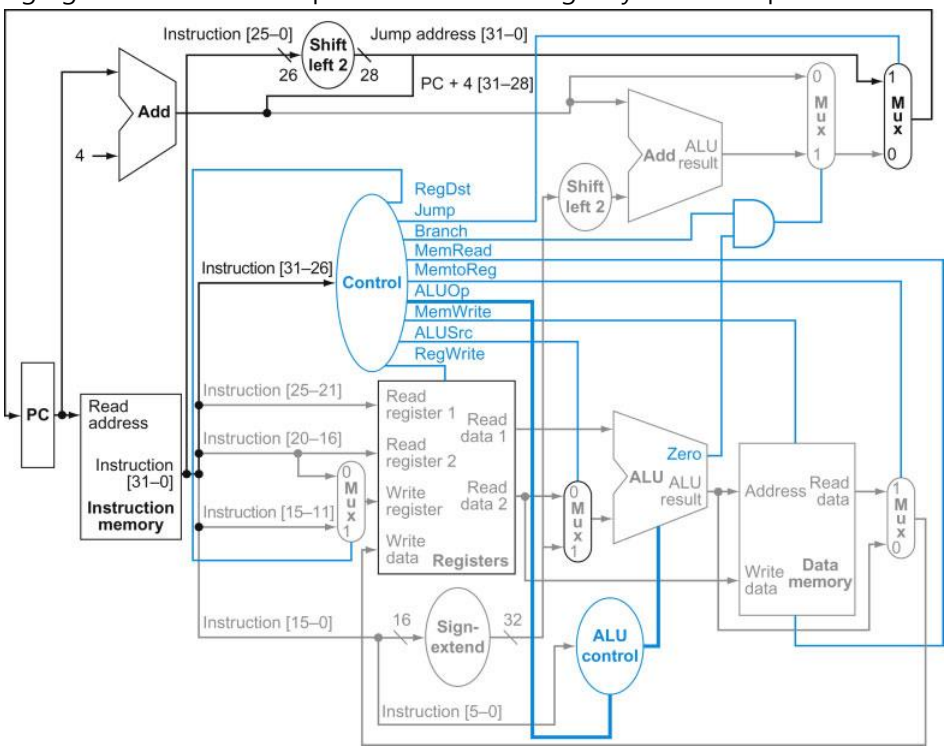

Figure 1 - The single cycle CPU datapath and control path

Table 1 – Instance name of top module

Instance name	Description
SC_CPU	Top module. Single-Cycle CPU
U0_PC	Program counter
U1_IM	Instruction memory
U2_RF	Register file
U3_SEU	Sign Extension Unit
U4_ALU	Arithmetic Logic Unit
U5_MULT	Multiplier Logic Unit
U6_DM	Data memory
U7_CTRL	Control unit -MainControl, MyControl

2. Assignment

MIPS Instructions **LW, SW, ORI, ADD, SUB, J, LUI, BREAK, LLO, LHI** are already implemented. You have to add the following MIPS Instructions:

XORI, SLT, SUBU, SRA, MULTU, MFLO, SB, LHU, BLEZ, JR

2.1 PLA Unit

- Module name: **PLA_AND.txt, PLA_OR.txt**

The AND-plane in a PLA detect specific input conditions. In the microprocessor instruction decoder, the AND-plane activates a line to indicate that a specific instruction is fetched. The OR-plane in a PLA generates designed output signals. In the control block of the microprocessor, the OR-plane generates control signals for datapath to form a specific path to perform the operation of the fetched instruction. **The endianness is little endian.** Before designing the PLA, you must **find the part of the CPU implementation where the control signal affects**.

Table 2 – Consecutive Instruction Decoding Configuration in PLA_AND.txt

Port name	Classification	Bit	Description
Op	Input	6-bit	Op code
Func	Input	6-bit	Function code
Regimm	Input	5-bit	Register Immediate code (RT)

*** For the MIPS opcodes, Function codes, and Control codes, See the attached file.**

Table 3 – Consecutive Control Signal Configuration in PLA_OR.txt

Port name	Classification	Bit	Description
RegDst	Output	2-bit	Register Control signal
RegDatSel	Output	2-bit	Register Write Data Selection signal
RegWrite	Output	1-bit	Register Control signal
SEUmode	Output	1-bit	Extender Control signal
ALUsrcB	Output	2-bit	ALU Input Selection signal
ALUctrl	Output	2-bit	ALU Control signal
ALUop	Output	5-bit	ALU Operation Control signal
DataWidth	Output	3-bit	Memory Data Control signal
MemWrite	Output	1-bit	Memory Control signal
MemtoReg	Output	1-bit	MEM/ALU Selection signal
Branch	Output	3-bit	Branch Address Control signal
Jump	Output	2-bit	Jump Address Control signal

*** The last 5 bits are reserved. Set them to xxxxx.**

The following figure depicts how the description in the PLA_AND matches the description in the PLA_OR. That it, the same line number means the same instruction. The format of each file is binary text (0, 1, and x) and the order is shown in the above tables. The character "_" can be used as a separator, and the simulator ignores it.

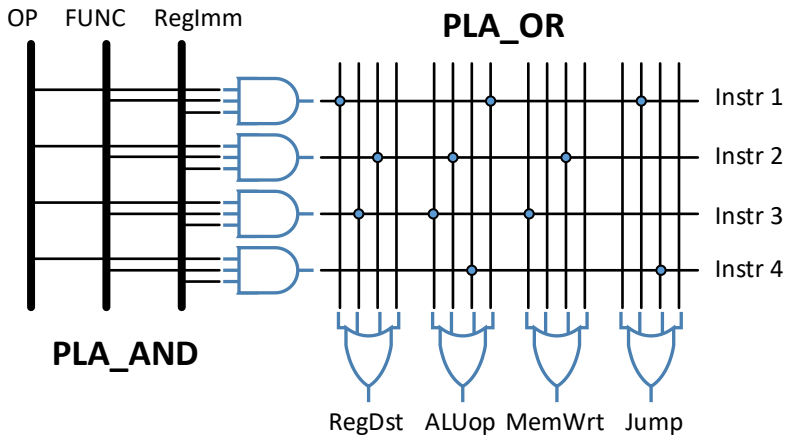

Figure 2 - The conceptual diagram of PLA

*** The simulator stops right after it executes the "break" instruction or when the cycle time reaches the number in the file "tb_cycle_limit.txt".**

*** The "mem_dump.txt" and "reg_dump.txt" show the data in 32bit word format which means first left-most byte has the highest address.**

3. 결과 Report (표지제외 최소 2장)

■ 문제의 해석 및 해결 방향

- 10개의 각 명령어에 대해 기능과 동작을 모두 상세히 설명할 것
  - 기능은 datasheet를 참조할 것
  - 동작에 대해서는 각 명령어의 datapath를 그림 1을 이용하여 그릴 것
- 실험 내용에 대한 설명

  ex) 자기가 구현한 하드웨어 구성, 동작 및 특징
- 문제점 및 개선점 기술
  - 기존 기능을 어떻게 이용하여 명령을 구현하였는지
  - 새로운 방법을 제시할 시 추가 점수

■ 설계 의도와 방법

- 구현한 Single Cycle CPU 블록도
- 전체 testbench 비교 분석
- 코드 주석 필수
- 시뮬레이션 결과와 예상 결과 비교 분석

4. 결과 Report Submission

■ Soft copy

- Due data: **4월 17일(수) 23:59 까지 (딜레이 받지 않음.)**
- 결과 Report(pdf)와 프로젝트 폴더를 압축하여 U-campus에 upload.
- **압축 파일명 양식: 학번_이름_Project_1.zip**
  - **ex) 2099722000_홍길동_ Project_1.zip**

5. 공지 사항

■ 본 프로젝트는 개인 프로젝트임으로 문제 해결을 위한 협력은 금함.

■ 부정행위 시 학칙에 의해 처벌될 예정

명령어의 정상적인 동작을 확인하는 실험 방법

M_TEXT_SEG.txt의 첫 5개 명령어를 수정하여 결과를 확인

1. 첫 4개의 명령어로 register 2개를 설정

lui $2, 0x1234	: **00111100 00000010 00010010 00110100**
ori $3, $2, 0x5678	: **00110100 01000011 01010110 01111000**
lui $4, 0x1122	: **00111100 00000100 00010001 00100010**
ori $5, $4, 0x3344	: **00110100 10000101 00110011 01000100**

2. 5번째 명령어에 test하고 싶은 명령어를 설정하여 실행결과를 확인 (뒤의 숫자는 결과 확인이 용이한 숫자로 변환하여 사용)

정상적인 명령어 동작의 확인을 위해서

- ALU의 결과,
- RF (Register file)의 write 결과,
- PC의 write 결과

등을 확인하여 예상하는 값이 write되는지 clock cycle이 rising하기 직전의 값을 확인

- 입력 data 값,
- Write enable 값