Lecture #10: Lab 3

The Infamous EE183 Pipelined Processor

Paul Hartke

Phartke@stanford.edu

Stanford EE183

May 8, 2002

Lab Stuff

- Lab #2 due ***TODAY*** at midnight.
- Paul in lab after class and in the lab at 11pm or so for demos.
- Any questions?

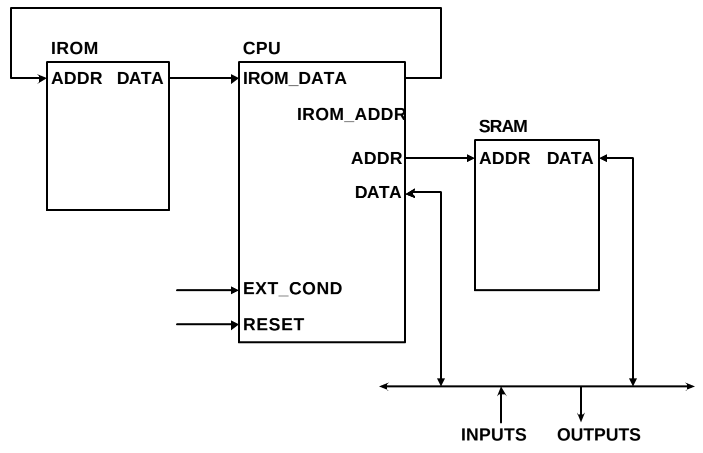
Processor Overview

- 12-bit RISC Microcontroller
  - What would having only 8 bits mean for the memory architecture?
- 4 or 8 General Purpose Registers
  - 4 back in the days when we had a small FPGA ☺
- 43 Instructions
- 3 operand instructions
- 4 stage pipeline
- Register indirect addressing mode
  - What does this mean?

Motivation

- Illustrate many datapath and control issues already discussed in class
- Complex enough to be "interesting"
- Simple enough to complete in 2 weeks
- Pipelining is an important technique in digital design
- ***Exciting!*** Tell your friends and look cool at dinner parties

Processor Overview



Instruction Set Architecture (ISA)

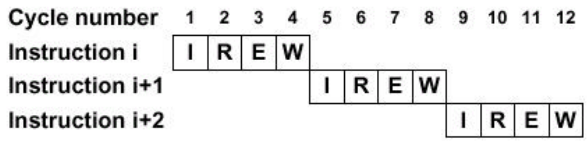
- 8 General Purpose Registers
- ALU Instructions
  - 28 Instructions
  - 3 operands
- Control Transfer Instructions
  - 12 Instructions
  - Conditional/Unconditional branches
- Memory Instructions
  - 2 instructions
  - Load/Store

Instruction Formats

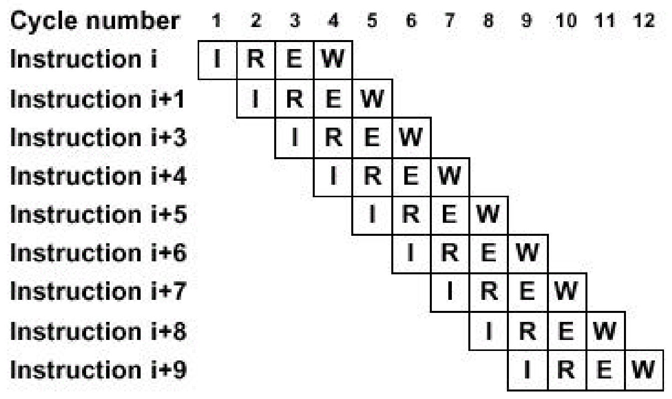
- (Save trees and look at the Lab 3 handout)

Instruction Execution Steps

- 4 Step Sequence
  - Step I Fetch instruction from Instruction Memory
  - Step R Read operands from registers (A, B)
  - Step E Execute instruction, set condition codes
  - Step W Write results to register C
- One stage per step
- Each instruction goes through all four stages
  - Assume each stage takes one clock cycle

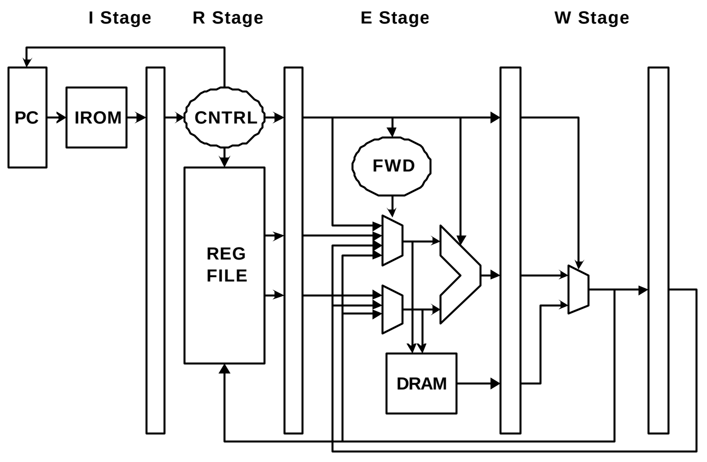


Pipeline



Cycle number	1	2	3	4	5	6	7	8	9	10	11	12
Instruction i	I	R	E	W								
Instruction i+1		I	R	E	W							
Instruction i+3			I	R	E	W						
Instruction i+4				I	R	E	W					
Instruction i+5					I	R	E	W				
Instruction i+6						I	R	E	W			
Instruction i+7							I	R	E	W		
Instruction i+8								I	R	E	W	
Instruction i+9									I	R	E	W

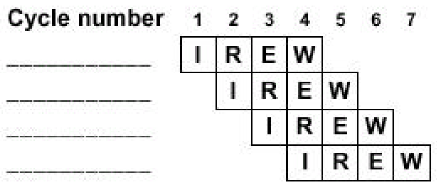
Pipelined CPU Block Diagram



Bypassing/Forwarding

- Given the following code fragment

  ADD R1, R2, R3
  SUB R4, R1, R5
  XXX
  YYY
- What's going on in the pipeline?

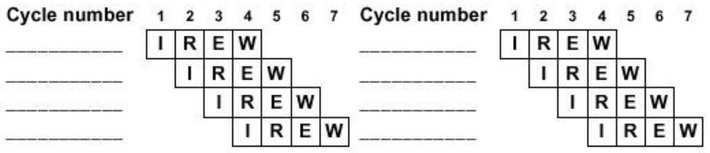


- How many different types of data hazards are there?

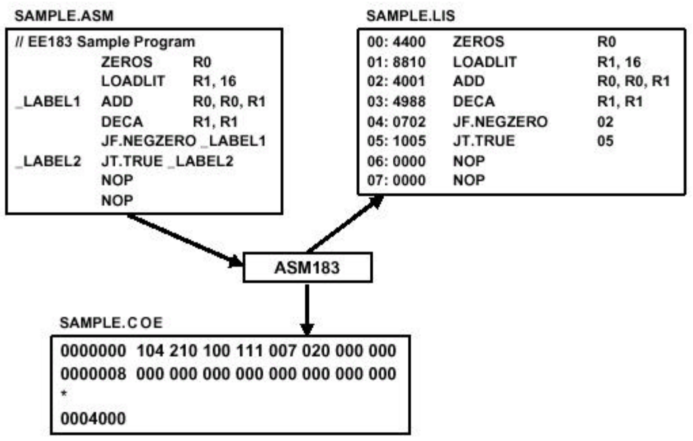
Control Transfer

- Code fragment

  00 ADD R1, R2, R3
  01 JT.ZERO _taken
  02 SUB R4, R5, R6
  03 AND R7, R8, R1
  …
  _taken 11 NOR R7, R8, R1
- Branch Taken vs. Branch Not Taken



ASM183 (Assembler)



SAMPLE.ASM
```
// EE183 Sample Program
          ZEROS      R0
          LOADLIT    R1, 16
_LABEL1   ADD        R0, R0, R1
          DECA       R1, R1
          JF.NEGZERO _LABEL1
_LABEL2   JT.TRUE _LABEL2
          NOP
          NOP
```

SAMPLE.LIS
```
00: 4400   ZEROS        R0
01: 8810   LOADLIT      R1, 16
02: 4001   ADD          R0, R0, R1
03: 4988   DECA         R1, R1
04: 0702   JF.NEGZERO   02
05: 1005   JT.TRUE      05
06: 0000   NOP
07: 0000   NOP
```

SAMPLE.COE
```
0000000  104 210 100 111 007 020 000 000
0000008  000 000 000 000 000 000 000 000
*
0004000
```

What do you get?

- Lab 3 Verilog
  - A lot of verilog given
  - Look through ALL of it
    - Some are not instantiated in the Lab 3 schematic
      - –e.g. **boolean.v**
- ASM183
  - Perl assembler
  - Perl Handout
    - How many already know perl?