KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
COLLEGE OF COMPUTER SCIENCES & ENGINEERING

COE 205 Computer Architecture & Assembly Language
Course Project -Term 112

Single Bus Processor Design

Due date: Wednesday, May. 16th, 2012

1. Introduction:

The aim of this project is a practical introduction to the architecture of the CPU. It is a straightforward application of the material covered in the lectures. The project consists on designing a basic CPU using the logisim software tool. The software is a very basic and handy tool that comprises the basic design elements of a digital system.

2. CPU Features:

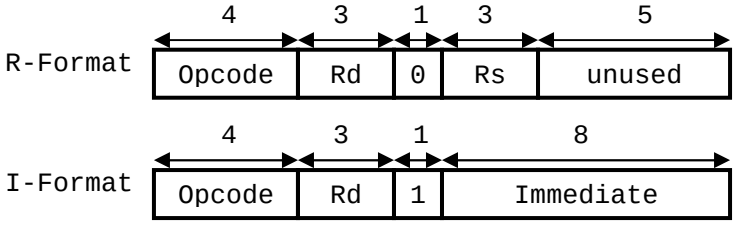
The CPU you are required to design is a 16-bit CPU, that has the following features:

- 16-bit data bus
- 16 bits address bus
- 8 16-bits registers organized as a register file (small memory)

- Instruction Register (IR)
- Memory Address Register (MAR)
- Memory Data Register (MDR)
- Program Counter (PC)
- Flags Register containing: the flags Zero (Z), Negative (N), Overflow (V) and Carry (C)

3. Instruction Format:

The CPU has two types of instructions; the R tye and the I type. When both operands are registers the instrcution is of the R type. When the second operand is a constant, the one is instrcution is of the I type. The instructions have the the following formats



The CPU has 3 instruction categories:

i. Arithmetic instructions:: Rd ← Rd op (Rs or Immediate)

Opcode	Description	Example	Register Transfer	ZF
0000	ADD	ADD Rdst, Rsrc	Rdst ← Rdst + Rsrc	A
0001	SUB	SUB Rdst, Rsrc	Rdst ← Rdst – Rsrc	A
0010	INC	INC Rdst	Rdst ← Rdst + 1	A
0011	DEC	DEC Rdst	Rdst ← Rdst – 1	A

ii. Logic instructions:

Opcode	Description	Example	Register Transfer	ZF
0100	AND	AND Rdst, Rsrc	Rdst ← and(Rdst,Rsrc)	A
0101	OR	OR Rdst, Rsrc	Rdst ← or(Rdst,Rsrc)	A
0110	XOR	XOR Rdst, Rsrc	Rdst ← xor(Rdst,Rsrc)	A
0111	NOT	NOT Rdst	Rdst ← not(Rdst)	A

iii. Shift and Rotate Instructions

Opcode	Description	Example	Register Transfer	ZF
1000	SHL	SHL Rdst, Rsrc	Rdst ← Rdst << Rsrc	A
1001	SHR	SHR Rdst, Rsrc	Rdst ← Rdst >> Rsrc	A
1010	ROL	ROL Rdst, Rsrc	Rdst ← Rdst << Rsrc	A
1011	ROR	ROR Rdst, Rsrc	Rdst ← Rdst >> Rsrc	A

iv. Data Transfer and Memory access instructions use R format. Load (Rd ← Mem[Rs]) and Store (Mem[Rs] ← Rd). The address is always contained in a register.

Opcode	Description	Example	Register Transfer	ZF
1100	MOV	MOV Rdst, Rsrc	Rdst ← Rsrc	U

v. Branch instructions

Opcode	Description		Example	Register Transfer	ZF
1101	JMP	Branch unconditional	JMP Rdst	IP ← Rdst	U
1110	JE, JZ	Branch if R1= R2	JZ Rdst	If (ZF=1) IP ← Rdst	U
1111	JG	Branch if R1> R2	JG Rdst	If (ZF=0) IP ← Rdst	U

Note: ZF = Zero Flag, A= affected, U = unaffected

4. CPU Design

The CPU has the following set of instructions:

(i) Design a **single-bus data path** for this CPU. Clearly illustrate all design details and all the required control signals. The CPU has only a Zero Flag (ZF) affected by the executed instructions as indicated in the above tables. Assume that you can only use the following combinational logic blocks in addition to basic gates like AND, OR, NOT, XOR, MUX and Tri-state Buffers:

- A 16-bit **Adder** that has the inputs Y[15:0], the data path and Cin, and produces the Sum in Z[15:0] and Cout.
- An 16-bit **Shifter** that has the inputs Y[15:0], for specifiying the input to be shifted, the amount of shift to be performed is specified from the data path. ShiftDir determines whether to shift left (ShiftDir=0) or shift right (ShiftDir=1). The shifted operand is produced on the output Z[15:0].

Design the data path to minimize the number of clock cycles needed for the execution of the specified instructions. Show the impact on the ZF in your design

(ii) Write the minimum number of control steps required for fetching an instruction from memory for this CPU, assuming asynchronous memory transfer, and knowing that an instruction is one word in size.

(iii) Show the block diagram of the **hardwired control unit** organization for this CPU indicating all the necessary components and signals. Clearly indicate the size of the various components. Derive the equations for the END signal and all the signals that control register R1.

(iv) Show the block diagram of the **microprogrammed control unit** organization for this CPU indicating all the necessary components and signals. Clearly indicate the size of the various components. Show the format and size of the control word. Show the microroutine of the JNZ instruction (show only the bits set to 1). Assume that the fetch microroutine is at address 0 and the address of JNZ microroutine is at address E. What is the **size of the control store** needed?

(v) **Memory:** Your processor will have separate instruction and data memories with 2^{12} = 4096 words each (this is the maximum that can be supported under the current version of Logisim). Each word is 16 bits or 2 bytes. Memory is *word addressable*. Only words (not bytes) can be read and written to memory, and each address is a word address. This will simplify the processor implementation. The PC contains a word address (not a byte address). Therefore, it is sufficient to increment the PC by 1 (rather than 2) to point to the next instruction in memory.

5. Plan

The following might be of some help designing your CPU.

1. ALU design
2. Data Path Design
   - Register Transfer
   - Register Transfer Timing
   - Single Bus CPU Design
3. Control Unit Design
   - Hardwired Control
   - Microprogrammed Control
4. CPU Design Testing

.

Program Execution

The program will be loaded and will start at address 0 in the instruction memory. The data segment will be loaded and will start also at address 0 in the data memory. You may also have a stack segment if you want to support procedures. The stack segment can occupy the upper part of the data memory and can grow backwards towards lower addresses. The stack segment can be implemented completely in software.

To terminate the execution of a program, the last instruction in the program can jump or branch to itself indefinitely.

Testing

To test your CPU, implement the selection sort procedure given in class along with Max procedure. Use this procedure to sort an array of 8 words of your choice. Then, write a second program that tests each of the remaining untested instructions to demonstrate their correct operation. Finally, write a third program that tests that your CPU can handle properly data and control hazards. Convert your programs into machine instructions by hand and load them into the instruction memory starting at address 0.

WARNING

Although Logisim is stable, it might crash from time to time. Therefore, it is best to save your work often. Make several copies and versions of your design before making changes, in case you need to go back to an older version

Project Report

The report document must contain sections highlighting the following:

1 – Design and Implementation

- Specify clearly the design giving detailed description of the datapath, its components, control, and the implementation details (highlighting the design choices you made and why, and any notable features that your processor might have.)
- Provide drawings of the component circuits and the overall datapath.
- Provide a complete description of the control logic and the control signals. Provide a table giving the control signal values for each instruction. Provide the logic equations for each control signal.
- Provide a complete description of the forwarding logic, the cases that were handled, and the cases that stall the pipeline, and the logic that you have implemented to stall the pipeline.
- Provide list of sources for any parts of your design that are not entirely yours (if any).
- Carry out the design and implementation with the following aspects in mind:
  - Correctness of the individual components
  - Correctness of the overall design when wiring the components together
  - Completeness: all instructions were implemented properly, detecting dependences and forwarding was handled properly, and stalling the pipeline was handled properly for all cases.

2 – Simulation and Testing

- Carry out the simulation of the processor developed using Logisim.
- Describe the test programs that you used to test your design with enough comments describing the program, its inputs, and its expected output. List all the instructions that were tested and work correctly. List all the instructions that do not run properly.
- Describe all the cases that you handled involving dependences between instructions, forwarding cases, and cases that stall the pipeline.
- Also provide snapshots of the Simulator window with your test program loaded and showing the simulation output results.

3 – Teamwork

- This project is a team work project with a maximum of three students. Make sure to write the names of all the group members on the project report title page.
- Each group should assign a group leader that leads the conduction of the project, divided the project tasks among the team members. The group leader will submit a weekly progress report summarizing the project progress.
- Project tasks should be divided among the group members so that each group member contributes equally in the project and everyone is involved in all the following activities:
  - Design and Implementation
  - Simulation and Testing
  - Design and results reporting
- Clearly show the work done by each group member using a chart and prepare an execution plan showing the time frame for completing the subtasks of the project. You can also mention how many meetings were conducted between the group members to discuss the design, implementation, and testing.
- Students who **help** other team members should mention that to earn credit for that.

Submission Guidelines

All submissions will be done through WebCT.

Attach one zip file containing all the design circuits, the test programs source code and binary instruction files that you have used to test your design, their test data, as well as the report document.

Grading policy:

The grade will be divided according to the following components:

- Correctness: whether your implementation is working
- Completeness and testing: whether all instructions and cases have been implemented, handled, and tested properly
- Participation and contribution to the project
- Report organization and clarity