Middle East Technical University | Department of Computer Engineering

CENG 232

Logic Design

Spring '2016-2017

Lab 4

Due date: 7 May 2017, 23:59
No late submissions!

1 Introduction

In this laboratory, you will be implementing the **memory subsystem** and you will complete the implementation of **data subsystem**. You will also be dealing with timing with the help of delays. Because of that, the fpga part is excluded, which means that you won't be testing your implementations with physical boards. Testing with test benches will be enough. .

2 Lab Work

In this lab, you are expected to implement Memory subsystem and complete the implementation of Data subsystem. The specifications and implementation details are provided in the following sections.

2.1 Memory Subsystem

In (Figure 1 - a) the overal structure of the memory subsystem is given. Each memory address consists of 24 bits. Consequently, memory subsystem is an array of 2^{24} locations. Each location is one byte, 2^{20} is one MB, so the memory is a total of 16 MB's.

Memory can be accessed by bytes of 32-bit words, organized as indicated in (Figure 1 - b). Successive byte addresses differ by 1, with successive addresses allocated from least-significant to most-significant positions within a word. The address of a word is the address of the least-significant byte within the word.

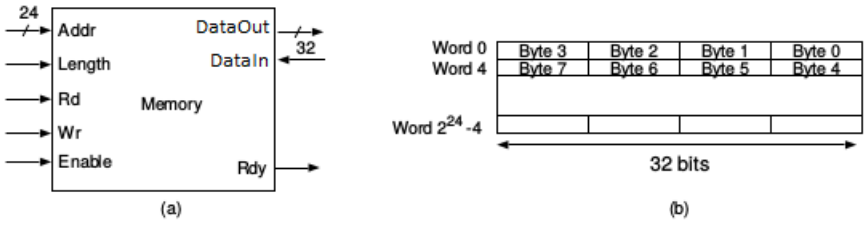

Figure 1: Memory Subsystem. a) External signals, b) Internal Organization

The operation of the memory is as follows:

- For the memory subsystem to be active, the Enable signal has to be set to 1;
- The subsystem responds to an event in either one of the control signals **Rd** or **Wr** (which are never active simultaneously);depending on which of these control signals is active, a memory read or a memory write operation is performed for an operand of the length specified in signal Length(0: byte, 1:word);
- The address of the memory location accessed is specified in signal **Addr**; in case of word accesses, the two lowermost bits of the address are ignored (they are assumed to be 00);
- The data is transferred through signal **DataOut** after 8ns delay;
- The completion of a memory operation is indicated by signal **Rdy**; this signal is set to 0 while the operation is in progress (with 200ps delay) and to 1 when the operation finishes (with 8ns delay).
- read followed by read, write followed by write, and write followed by read can be performed keeping signal **Enable** set to 1; for each operation, a pulse is required in the corresponding signal **Rd** or **Wr**;
- a read followed by a write requires that signal **Enable** is set to 0 and then set again to 1.

Note that the operation of the memory subsystem is not synchronized by a clock signal; instead, memory operations are performed in response to the memory control signals. An active control signal(**Rd** or **Wr**) initiates the operation of the module; signal **Rdy** is set to 0 with a 200ps delay meaning that the module is busy. Furthermore, the module does not accept further requests until the current memory access has completed.

The signature of the module is given below:

```
module Memory(
input [23:0] Addr,
input Length, //byte:0 vs. word:1
input Rd,
input Wr,
input Enable,
output reg Rdy,
input [31:0] DataIn,
output reg [31:0] DataOut
);
```

2.2 Data Subsystem

A network that implements the requirements of the data subsystem is depicted in (Figure 2). This network consists of a **register file** (RF) that contains the general-purpose registers, dedicated registers for **PC, CR and IR**, **an arithmetic-logic unit (ALU)**, as well as other supporting modules. Each of these modules has control signals that determine its operation. In the previous homework you had already implemented some of these (ALU, Register etc). This will be provided to you with the delays implemented in them. In this section you will implement the remaining parts of the data subsystem: **a)** register File (RF), **b)**switch and **c)**extender. In addition, you will implement the interconnections between all data subsystem components.

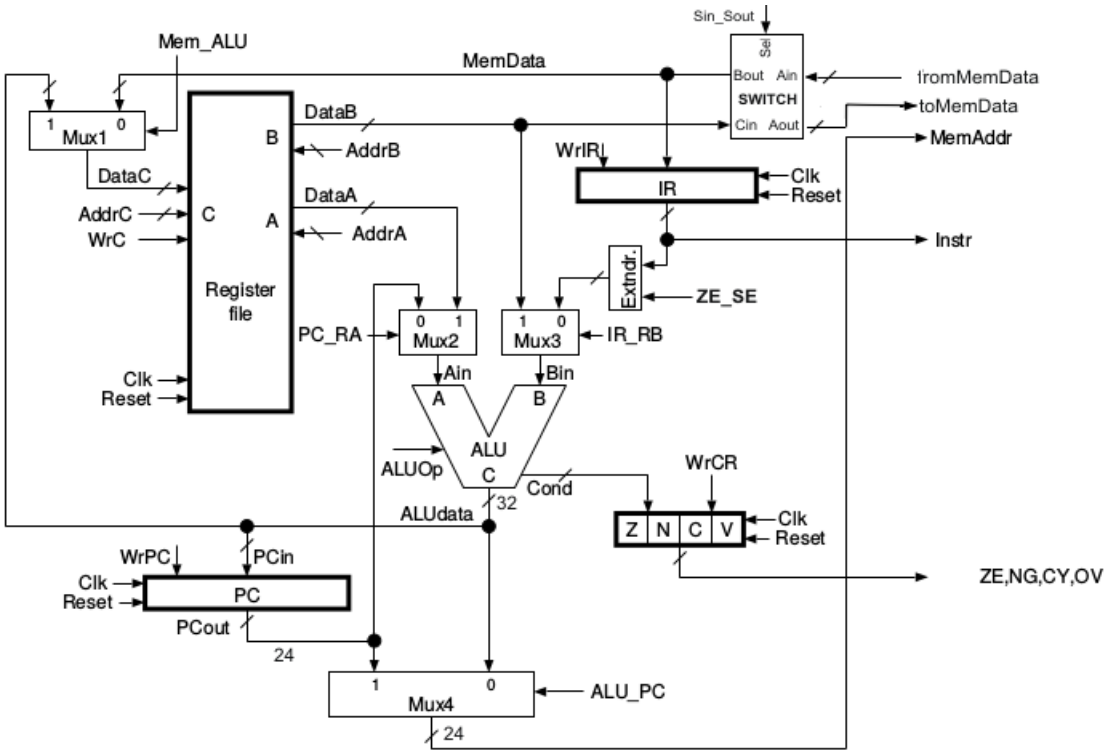

Figure 2: Implementation of data subsystem.

2.2.1 Register File

The register file (RF) contains 32 registers of 32 bits each. It has two **read ports** named A,B, which are used to read up to two operands required by an instruction, and one **write port** named C (WrC Synchronous) used to write the result. Each port has its own address (5 bits) and data signals (32 bits); the write port also has a control signal. As a result, two read operations and one write operation can be performed simultaneously. Each read port (A, B) delivers the data stored in the register selected by the corresponding address signal. When address of A or B changes data output of related address (A or B) is updated. The write port stores the data present at the DataC input into the register selected by the AddrC signal, synchronized with signal Clk. Both operations have a delay of **RF_delay** (4ns). All registers are set to zero when the signal **Reset** (Asynchronous) is active.

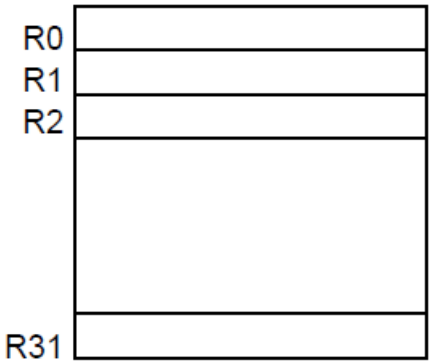

Figure 3: Register File

The signature of the module is given below:

```
module Reg_File(
input [4:0] AddrA,
input [4:0] AddrB,
input [4:0] AddrC,
output reg [31:0] DataA,
output reg [31:0] DataB,
input [31:0] DataC,
input WrC,
input Reset,
input Clk
    );
```

2.2.2 Switch

Switch module operates as follows:

- When selection value (**Sel**) is 0, after switch delay (500ps) it connects Ain to Bout.
- When **Sel** is 1, after switch delay (500ps) it connects Cin to Aout.

The module produces the output whenever one of its inputs change. The signature of the module is given below:

```
module Switch32(
input [31:0] A_in,
output reg [31:0] A_out,
output reg [31:0] B_out,
input [31:0] C_in,
input Sel
    );
```

2.2.3 Extender

The zero/sign-extender changes the left-most 16 bit of its 32-bit input according to its control signal **ZE_SE** (0: extends with zero, 1: extends according to sign) with a delay of 500ps.

The module produces the output whenever one of its inputs change. The signature of the module is given below:

```
module Extender(
input [31:0] X_in,
input ZE_SE,
output reg [31:0] X_out
    );
```

2.2.4 Dedicated Registers

The data subsystem contains dedicated registers for IR,PC and CR (32 bits, 24 bits and 4 bits, respectively). The dedicated registers are loaded whenever the corresponding control input is set to 1 (WrIR, WrPC and WrCR respectively), synchronized with signal **Clk**. The new value is delivered at the output after a register read delay (**Reg_delay**). The registers are set to zero whenever signal RESET is active.

2.2.5 Interconnections

Here are some characteristics of interconnections of the modules of data subsystem:

- The connection to memory consists of the data bus (**MemData**) and the address bus (**MemAddr**); the address corresponds to the lower 24 bits either from the ALU output (for accessing data) or from register PC (for fetching instructions);
- The RF write port (DataC) is shared by the paths from memory and from the ALU;
- The RF read port B is shared by the paths to memory and to ALU input B;
- ALU input A is shared by RF and PC;
- ALU input B is shared by RF and IR (the 16-bit fields from IR are sign-extended or zero-extended)

The signature of the all data subsystem module is given below:

```
module Data_Subsystem(
output  [23:0] MemAddr,
input [31:0] fromMemData,
output  [31:0] toMemData,
output  [31:0] Instr,
output  ZE,
output  NG,
output  CY,
output  OV,
input [4:0] AddrA,
input [4:0] AddrB,
input [4:0] AddrC,
input [3:0] ALUop,
input WrC,
input WrPC,
input WrCR,
input WrIR,
input Mem_ALU,
input PC_RA,
input IR_RB,
input ALU_PC,
input ZE_SE,
input Sin_Sout,
input Clk,
input Reset
    );
```

3 Deliverables

- Implement the modules in 5 files (`Memory.v` ,`Switch32.v` ,`Extender.v`, `Data_Subsystem.v` and `Reg_File.v`) and compress them into a single file named `lab4.zip`. **Do not submit your testbenches or additional files.**
- It is your responsibility to write a testbench for the modules. You may share your testbenches on the newsgroup.

- Submit the zipped file through the COW system before the given deadline. The homework is supposed to be done with your group partner. Make sure both of you take roles in implementation of the project. Any kind of inter-group cheating is not allowed.
- Use the newsgroup metu.ceng.course.232 for any questions regarding the homework.