**Review Ch.3-4**

**Arithmetic and Datapath**

1. Describe the algorithm that performs the multiplication of two binary integers.

Repeat the following steps 32 times:

If the multiplier is 0 than we shift the multiplicand register left 1 bit, then we shift the multiplier right 1 bit and we go back in the loop.

If the multiplier is 1 than we add the multiplicand to the current product and we store the result in the Product register, then we shift the multiplicand register left 1 bit, we shift the multiplier right 1 bit, and we go back in the loop.

1. (Skip this question) Perform the division of 100111 by 100.
2. What is the difference between single and double precision?

Single precision is represented in 32-bit and Double precision is represented in 64-bit. The format for single precision allows 8 bits for the exponent and 23 bits for the fraction. Double precision format allows 11 bits for the exponent and 52 bits for the fraction.

1. **Use the IEEE 754 standard for the representation of the following numbers:**
   1. -35.25

In binary the number is -100011.01, which is rewritten in normalized scientific notation as -1.0001101 x 25 . After **adding the bias to 5 we get 5+127=132** which in binary corresponds to 10000100

The single precision binary machine representation is 1 1000 0100 0001 1010 0000 0000 0000 000

* 1. 125.75

Binary conversion 1111101.11 – Normalized Scientific Notation 1.11110111 x 26 - Exponent+ Bias = 6+127=133 which in binary is 1000 0101

0 1000 0101 111 1011 1000 0000 0000 0000

1. What is the overflow? What is the underflow? Give an example of overflow and underflow. When does it occur?

Overflow occurs when the result is too large to be represented in 32 bits. Underflow occurs when the result of a floating point operation is smaller in magnitude (that is, closer to zero) than the smallest value representable as a normal floating point number.

Example of overflow: 11001 + 11111 = generates overflow

The resulting number is seen as a negative number since it starts by 1.

Example of underflow: if the exponent part can represent values from −127 to 127, then a result with absolute value less than 2−127 causes underflow.

1. Perform the addition of these numbers:

1.110001 \* 2-7 + 1.0001 \* 25 + 1.11110 \* 21 = 10010.11110001110001 \* 21

1. **Perform the multiplication of these numbers:**

1.110001 \* 2-4 +1.110 \* 21 = 11.000100111 \* 2-3

1. ~~(Skip this question) What is the benefit of using the guard and round in floating point arithmetic?~~

~~Guard is a two extra bits kept on the right during intermediate calculation of floating point numbers to improve rounding accuracy.~~

~~Rounding is used to fit the floating point number to the given format.~~

1. What is the job of a multiplexor? What is the value assigned to the data selector of a multiplexor if we want to output the signal carried by the fourth wire entering in input the multiplexor?

The job of a multiplexor is to select an input line to send in output. The selected input line will depend on the value of the data selector line.

When the value of the data selector line is 11, which represents the number 3 in binary, the value of the fourth input line will be sent in output. (Remember that the first line is selected with the value 00. So 11 is the fourth line.)

1. Consider the simple single-cycle datapath described in your book. How many functional units (or datapath elements if you prefer) are part of the implementation of the datapath?

The Instruction Memory, the Register, the ALU, the Data Memory, and multiple adders.

1. What is the purpose of the Data Memory in the single-cycle implementation?

It is used for load and store instruction.

1. A state element is also called a …………… element.

… memory element, such as a register or a memory cell.

1. What is an edge-trigger clocking?

A clocking scheme in which all state changes occur on a clock edge.

1. What is the need for a clock?

To ensure predictability and define when signals can be read and when they can be written.

1. **Describe the path taken by the load instruction in a** **single-cycle implementation scheme. (answer the same question for all the other types of instructions)**

**See Figure 4.20 for the load, Fig 4.19 for the R-type instruction, Fig 4.21 for the beq instruction, Fig 4.24 for the jump instruction.**

1. Which instructions use the Sign Extend unit in the single-cycle implementation scheme?

All the I format instructions.

1. When is the MemtoReg asserted in a single-cycle implementation scheme? Give the name of an instruction that requires the assertion of the MemtoReg.

The MemtoReg control is asserted to 1 when the instruction wants to write the result retrieved from the Data Memory back to the Register. The load instruction requires the assertion of MemtoReg.

1. Explain the purpose of the control unit in a single-cycle implementation scheme.

The Control Unit sets the values of the control lines that permit the correct execution of the instruction.

1. **There are two possible 5bit chunks, i.e. [20-16] and [15-11] of the 32 bit instructions that are used to indicate the Write Register. A Mux is used to select the appropriate one. Describe in which case the [20-16] is used and in which case the [15-11] is used.**

When an R-format instruction is used, the 15-11 bit positions are used and control line RegDest is set to 1 to get from the Mux the appropriate destination register. When an I-format instruction is used, the 20-16 bit positions are used.

1. Why is a multi-cycle implementation scheme introduced?

To enhance performance by executing instructions in parallel in a manner that, when the resources are available, the next instruction is started before the previous instruction has finished.

1. What are the differences of the multi-cycle implementation scheme with respect to the single-cycle implementation scheme?

A single-cycle schema only allows for one instruction to be executed in a clock cycle. All the instructions have the same length and the next instruction has to wait until the current instruction is finished before starting.

A multi-cycle schema reduces the total time to complete a set of instructions, since each instruction is performed in multiple stages, each stage take a clock cycle, and the next instruction starts, if possible, after the completion of the first stage of the previous instruction. This causes the parallel execution of instructions which minimizes the total execution time.

1. **The time to execute a load in a multi-cycle implementation scheme is slower than the time to execute a load in a single-cycle implementation scheme. Why is that?**

**The load instruction is the longest instruction both in single-cycle implementation and in the multiple-cycle implementation. In the multiple-cycle implementation, in order to execute in a proper way the load instruction, information must be carried between stages in registers that must be written at the end of each stage and read in order to be available at the beginning of the next stage. This additional activity adds an overload that slows the overall time of the load instruction in the multiple-cycle implementation.**

1. *Trace a store instruction on the multicycle implementation scheme. What are the values of the control signals in each of the cycles?*

*See Figure 4.36 for first and second stage, figure 4.39 for the 3rd stage, figure 4.40 for 4th and 5th stage.*

1. In which stage of the multicycle implementation, the following steps are performed?
   1. the Instruction[25:21]of the instruction is read and given in input to the ALU
   2. the Instruction[20:16]of the instruction is read and given in input to the ALU
   3. the PC and the (sign-extend(Instruction[15-0] << 2) is given in input to an adder;

In the EX stage of a branch instruction. The first two steps read t Execute the address calculation of a branch

1. Which is the shortest instruction in a multi-cycle implementation scheme?

The Jump instructions (J format instruction).

1. (Skip this question) Is it possible that the control signal PCWriteCond can be replaced by the PCSource[0] in the multi-cycle implementation scheme. If so, when? SKIP THIS
2. What is a “don’t care” symbol and when it is used? This answer is in the book not in the slides.

The don’t care symbol is used to indicate when the value of a control line is independent from the result of the executing instruction.

The symbol X is used as don’t care symbol.

1. **Consider the following code:**

**Repeat:lw $t1, 0($t2)**

**subi $t1, $t1, 3**

**sw $t1, 0($t2)**

**addi $t3, $t3, 12**

**bne $zero, $t3, Repeat**

**and simulate it on a 5-stages pipeline.**

**lw $t1, 0($t2) IF ID EX MEM WB**

***stall- data hazard***

**subi $t1, $t1, 3 IF ID EX MEM WB**

**sw $t1, 0($t2) IF ID EX MEM WB**

**addi $t3, $t3, 12 IF ID EX MEM WB**

**bne $zero, $t3, Repeat IF ID EX MEM WB**

1. What types of hazards are available in a pipelined implementation?

Structural hazard: when tow instruction want to sue the same resource

Data hazard: when an instruction want to use a data that is not ready yet

Control hazards: When a control instruction needs to delay to determine the proper instruction to fetch.

1. Consider the following code.
   1. How many cycles will be required to properly execute the following code if NO additional techniques are used to prevent or eliminate stalling? Specify also where each stall occurs.

add $t1, $t2, $t3

subi $t1, $t1, 3

lw $t1, 0($t2)

add $t3, $t3, $t1

bne $zero, $t3, Loop

* 1. How many cycles will be required to properly execute the previous code if forwarding or code optimization is used to minimize stalling? Show where forwarding, stalling, or instruction permutation occurred.

a)

IF ID EX MEM WB *(add)*

- - - - -

- - - - -

IF ID EX MEM WB *(subi)*

If ID EX MEM WB *(lw)*

- - - -

- - - -

If ID EX MEM WB *(add)*

- - - - -

- - - - -

IF ID EX MEM WB *(bne)*

add $t1, $t2, $t3

stall

stall

subi $t1, $t1, 3

STALL

STALL

lw $t1, 0($t2)

stall

stall

add $t3, $t3, $t1

bne $zero, $t3, Loop

15 CYCLES IN ALL

b)

IF ID EX MEM WB *(add)* -

IF ID EX MEM WB *(subi)*

IF ID EX MEM WB *(lw)*

- - - - -

IF ID EX MEM WB *(add)*

IF ID EX MEM WB *(bne)*

10 CYCLES IN ALL if forwarding is used.

1. What is forwarding and how is it realized?

Forwarding is a technique used in multi-cycle implementation to retrieve data produced in the pipeline before they are effectively stored in the proper location. Observe that an instruction needs the data in the Ex stage. In the case of data hazard, the data can be taken either from the EX/MEM of the previous instruction or from the MEM/WB register of two instructions before. Forwarding requires a check for dependency (dependency check is on pp306-311 of your book) and appropriate control lines that selected the appropriate data from the appropriate location.

1. Can forwarding eliminate any data hazard? Explain.

No. Forwarding cannot eliminate all the possible data hazards. For example, when an instruction that follows a *lw* instruction causes a data hazard, since the data is ready only in the MEM stage due to the length of the load instruction, a stall must be injected in the pipeline. In other situations, it is possible to eliminate the stalls caused by the data hazards via forwarding.

1. What is dynamic branch prediction and where is it used?

Dynamic prediction is a technique used to overcome control hazard. The idea is to keep track if a branch has been taken or not and use that information to predict if the next branch will be taken or not. If the previous branch has been taken, then a prediction that the next branch will be taken is used. If the previous branch has not been taken, then a prediction that the next branch will not be taken is used. Since in regular loops, this prediction causes missing the correct prediction twice (first at the entry of the loop, then at the exit of the loop) a 2-bit prediction is used to improve prediction. In 2-bits prediction the prediction is changed only if the previous prediction has been missed twice. In regular loops, now we miss once instead of 2 times.