NYU Tandon School of Engineering Fall 2022, ECE 6913

**Homework Assignment 5** *Instructor: Azeez Bhavnagarwala,* email: [ajb20@nyu.edu](mailto:ajb20@nyu.edu) *Course Assistants*

Varadraj Kakodkar (vns2008), Kartikay Kaushik (kk4332), Siddhanth Iyer (si2152), Swarnashri Chandrashekar (sc8781), Karan Sheth (kk4332), Haotian Zheng (hz2687), Haoren Zhang (kk4332), Varun Kumar (vs2411)

**Homework Assignment 4** [released Wednesday October 23rd 2022] [due Wednesday November 2nd by 11:59PM]

You *are allowed* to discuss HW assignments with anyone. You are *not allowed* to share your solutions with other colleagues in the class. Please feel free to reach out to the Course Assistants or the Instructor during office hours or by appointment if you need any help with the HW.

Please enter your responses in this Word document after you download it from NYU Classes.

*Please use the Brightspace portal to upload your completed HW.*

*.*

1. In this exercise, we examine in detail how an instruction is executed in a single-cycle datapath. Problems in this exercise refer to a clock cycle in which the processor fetches the following instruction word: **0x00c6ba23**.

Note:

0x00c6ba23 = 0000 0000 1100 0110 1011 1010 0010 0011

The core instruction format for a single-cycle data path is:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Immediate[11:5] | rs2 | rs1 | Funct3 | Immediate[4:0] | opcode |
| 7 bits | 5 bits | 5 bits | 3 bits | 5 bits | 7 bits |

0000 0000 1100 0110 1011 1010 0010 0011.

Since this is an s-type, the instruction is- sd x12, 20(x13)

* 1. What are the values of the ALU control unit’s inputs for this instruction?

ANSWER:

The ALUop bits for sd instruction is 00

The Control bits for ALU are 0010.

In ALU, for the ‘sd’ instruction, the ‘add’ operation will be executed.

Therefore we will add the offset in the immediate 5 bit field (20) to the base address (in rs1).

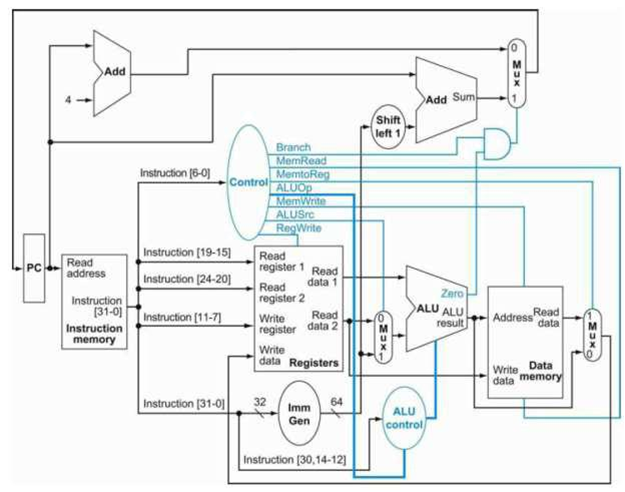
* 1. What is the new PC address after this instruction is executed? Highlight the path through which this value is determined.

ANSWER:

The new PC address instruction that is executed is PC+4.

(the sd instruction will not take any branch)

The image below has been taken from the fig 4.17 in textbook.



* 1. For each mux, show the values of its inputs and outputs during the execution of this instruction. List values that are register outputs at Reg [xn].

ANSWER:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mux | Control input | Input 1 | Input 2 | output |
| ALUsrc | 1 | rs2 | 0x00..014 | 0x0..014 |
| PCsrc | 0 | PC +4 | Offs s112 | PC+4 |
| MemtoReg | irrelevant | rs1+0x..14 | unknown | unknown |

* The ALUsrc control input value is 1. From the sign extended unit the input(0x0..014) is taken to the mux. The decimal number 20 corresponding to the sign extended offset is 0x00.014
* Since the mux is only asserted to the branch instruction the PCsrc control input value will be ‘0’and its output will be PC+4.
* The memtoReg value is irrelevant because, the mem write control bit is ‘1’ and the reg write control bit is ‘0’according to the 7-bit opcode of the instruction. Therefore the undefined data will not be used in the register
  1. What are the input values for the ALU and the two add units?

ANSWER:

The inputs to the ALU are:

1. rs1(Reg[x13]
2. 0x00..014

The inputs to the branch adder are:

1. Ps
2. 0x00..014

The inputs to the PC adder are:

1. Pc
2. 4
   1. What are the values of all inputs for the register’s unit?

ANSWER:

***rs1***:

The read registers 1’s input port has a 5-bit instruction field from the bits 19 -15 which stores the register x13.

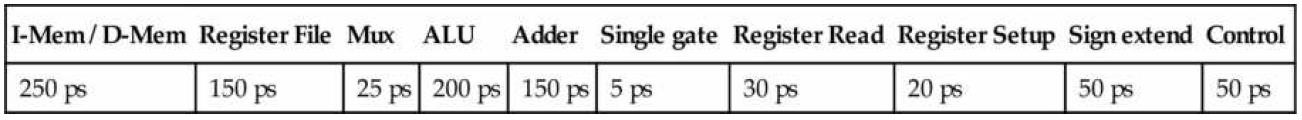
***rs2:***

The read register 2’s input port has a 5-bit instruction field from the bits 24-20 which stores the register x11.

***Write register:***

For sd instruction the Reg Write control signal is disabled. Therefore its input field is irrelevant

1. Problems in this exercise assume that the logic blocks used to implement a processor’s datapath have the following latencies:



*“Register read” is the time needed after the rising clock edge for the new register value to appear on the output. This value applies to the PC only. “Register setup” is the amount of time a register’s data input must be stable before the rising edge of the clock. This value applies to both the PC and Register File.*

* 1. What is the latency of an R-type instruction (i.e., how long must the clock period be to ensure that this instruction works correctly)?

ANSWER:

The following is the latency of an R-type instruction:

* The delay for reading a PC register is: 30 ps
* Instruction Memory: 250ps
* Reading a register file: 150ps
* Sign extended bit from immediate field of instruction: 25 ps
* The delay in ALU: 200ps
* From ALU, time to reach from Mux to register file: 25ps
* Setup time to write back for file registers: 20ps

Total = 700 ps

* 1. What is the latency of ld? (Check your answer carefully. Many students place extra muxes on the critical path.)

ANSWER:

The following is the latency for load instruction:

* The delay for reading a PC register is: 30 ps
* Instruction Memory: 250ps
* Reading a register file: 150ps
* Sign extended bit from immediate field of instruction: 25 ps
* The delay in ALU: 200ps
* Read data from Data Mem, whose address which is provided by ALU: 250 ps
* From ALU, time to reach from Mux to register file: 25ps
* Setup time to write back for file registers: 20ps

Total = 950ps

* 1. What is the latency of sd? (Check your answer carefully. Many students place extra muxes on the critical path.)

ANSWER:

The latency for the sd instruction is:

* The delay for reading a PC register is: 30 ps
* Instruction Memory: 250ps
* Reading a register file: 150ps
* Sign extended bit from immediate field of instruction: 25 ps
* The delay in ALU: 200ps
* Write data from r2 to data mem: 250ps

Total: 905

* 1. What is the latency of beq?

ANSWER:

The latency for beq:

* The delay for reading a PC register is: 30 ps
* Instruction Memory: 250ps
* Reading a register file: 150ps
* Sign extended bit from immediate field of instruction: 25 ps
* The delay in ALU: 200ps
* Delay of ‘zero’ from ALU and ‘branch’ from control unit: 5ps
* Mux for branch to PC: 25ps
* Write back time for PC register: 20ps

Total= 705ps

* 1. What is the latency of an I-type instruction?

ANSWER:

Latency for the I-type instruction is:

* The delay for reading a PC register is: 30 ps
* Instruction Memory: 250ps
* Reading a register file: 150ps
* Sign extended bit from immediate field of instruction: 25 ps
* The delay in ALU: 200ps
* Read data from Data Mem, whose address which is provided by ALU: 250 ps
* From ALU, time to reach from Mux to register file: 25ps
* Setup time to write back for file registers: 20ps

Total = 950ps

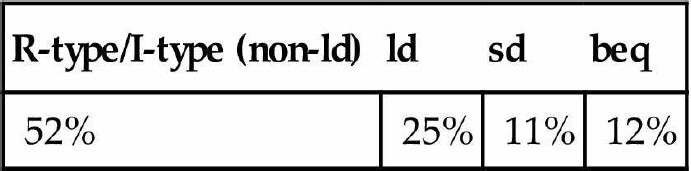
* 1. What is the minimum clock period for this CPU?

ANSWER:

Since it is the longest execution time across all instructions , the minimum clock period for this CPU is: 950 ps

1. **(a)** Suppose you could build a CPU where the clock cycle time was different for each instruction.

***3.a1*** What would the speedup of this new CPU be over the CPU presented in Figure 4.21 (in RISC-V text) given the instruction mix below? (assuming instruction latencies from the **problem 2)**



***ANSWER:***

= 700ps x 0.52 + 950ps x 0.25 + 905ps x 0.11 + 705ps x 0.12

== 785.6ps

= 1.21

**3 (b)** Consider the addition of a multiplier to the CPU shown in Figure 4.21. This addition will add 300 ps to the latency of the ALU, but will reduce the number of instructions by 5% (because there will no longer be a need to emulate the multiply instruction).

3.b1 What is the clock cycle time with and without this improvement?

Clock cycle time without improvement was : 950ps

Adding the multiplier will increase the clock cycle by 300ps

Therefore the new clock cycle time will be: 1250 ps

Clock cycle after adding the multiplier with 5% reduced number of instruction

= 0.95 x 1250

= 1187.5 ps

3.b2 what is the speedup achieved by adding this improvement?

ANSWER:

= 0.8

3.b3 what is the slowest the new ALU can be and still result in improved performance?

ANSWER:

The addition of the multiplier in the new ALU will reduce the number of instructions by 5% i.e. it improved the cycle time by 0.05 \* 950ps = 47.5 ps.

But, this improvement also resulted in the addition of a penalty of 300 ps which is a lot more than the reduction in the cycle time.

Therefore, the new ALU cannot be any slower than it is to improve the performance of the new CPU.

**3 (c)** When processor designers consider a possible improvement to the processor datapath, the decision usually depends on the cost/performance trade-off. In the following three problems, assume that we are beginning with the datapath from Figure 4.21, the latencies from **Problem 2** in this assignment, and the following costs:

Suppose doubling the number of general-purpose registers from 32 to 64 would reduce the number of ld and sd instruction by 12%, but increase the latency of the register file from 150 ps to 160 ps and double the cost from 200 to 400. (*Use the instruction mix [from 3(a) above] and ignore the other effects on the ISA*)

***3.c1*** What is the speedup achieved by adding this improvement?

ANSWER:

Since the number of ld and sd instructions have been reduced by 12%,

The total number of instructions will be reduced by 12%

Therefore the instruction count is

Therefore, the total number of instructions now = (100 - 4.32)%

= 95.68%(of the total instruction)

Also the latency of the register file will be increased from 150ps to 160ps

Hence,

The new cycle time = (950 + 10) \* 0.9568

= 918.528 ps

Speedup = 950 / 918.528

= 1.034

=3.4% increase in performance

***3.c2*** Compare the change in performance to the change in cost.

ANSWER:

|  |  |  |
| --- | --- | --- |
| ***Component*** | ***Original CPU*** | ***New CPU*** |
| PC | 5 | 5 |
|  |  |  |
| I-Mem | 1000 | 1000 |
|  |  |  |
| Register File | 200 | 400 |
|  |  |  |
| ALU | 100 | 100 |
|  |  |  |
| D-Mem | 2000 | 2000 |
|  |  |  |
| Sign Extend | 100 | 100 |
|  |  |  |
| Controls | 500 | 500 |
|  |  |  |
| Adders | 30 x 2 | 30 x 2 |
|  |  |  |
| Muxes | 10 x 3 | 10 x 3 |
|  |  |  |
| Single Gates | 1 | 1 |
|  |  |  |
| **Total** | **3996** | **4196** |
|  |  |  |

= 1.05

= 5%

Therefore, 5% increase in cost Performance increase of 3% for a cost increase of only 5% is a good design choice.

1. ***c3*** Given the cost/performance ratios you just calculated, describe a situation where it makes sense to add more registers and describe a situation where it doesn’t make sense to add more registers.

ANSWERS:

In the applications where the revenues are very performance sensitive, there will be a mere 3% improvement in performance and it could easily justify a 10% increase in cost.

These examples are generally encountered in HPC processors for the data center.

For applications where cost is the primary metric – IoT & embedded systems for example, it would not make sense to make it 10% more expensive to justify a performance improvement of 3%.

1. ld is the instruction with the longest latency on the CPU from Section 4.4 (in RISC-V text). If we modified ld and sd so that there was no offset (i.e., the address to be loaded from/stored to must be calculated and placed in rs1 before calling ld/sd), then no instruction would use both the ALU and Data memory. This would allow us to reduce the clock cycle time. However, it would also increase the number of instructions, because many ld and sd instructions would need to be replaced with ld/add or sd/add combinations.
   1. What would the new clock cycle time be?

ANSWER:

When we modify the ld instruction such that the instruction would not use both the ALU and Data memory, the latency of the modified ld instruction will be:

* The delay for reading a PC register is: 30 ps
* Instruction Memory: 250ps
* Register File: 150 ps
* Read data from Data Memory of the address provided by ALU: 250 ps
* Mux for Write back to Register for the result from Data Memory: 25 ps
* Write back setup time for Register File registers: 20 ps

Total latency for the modified id is: 725ps

With the modification of sd instruction such that the instruction would not use both the ALU and Data memory, the latency of the modified sd instruction will be:

* The delay for reading a PC register is: 30 ps
* Instruction Memory: 250ps
* Register File: 150 ps
* Write back to memory: 250ps

Total latency for the modified sd will be 680ps.

Since the ld instruction takes the longest time, the new cycle time would be 725ps

* 1. Would a program with the instruction mix presented *in Problem 2* run faster or slower on this new CPU? By how much? (For simplicity, assume every ld and sd instruction is replaced with a sequence of two instructions.)

ANSWER:

According to the instructions given, the ld and sd instructions will account for 36% of the total number of the instructions that are given.

Since we r modifying the instruction we will require 1 extra instruction for both ld and sd instructions,

Hence, the totally number of instructions will be 36% more compared to the original number of instruction.

=986

The new CPU will be 3.65% slower.

* 1. What is the primary factor that influences whether a program will run faster or slower on the new CPU?

ANSWER:

In , cycle time for the ld and the sd instruction Is improved and the instruction count is increased.

If the number of instructions is very small the CPU will perform faster but if the number of instructions are larger then the CPU will perform slowly.

Let,

The percentage of both ld and sd instructions when combined = x%

For the cpu to be faster,

Therefore if the percentage of ld/sd instructions is less than 30% of the total number of instructions then the new CPU will be faster or else it will be slower.

* 1. Do you consider the original CPU (*as shown in Figure 4.21 of RISC-V text*) a better overall design; or do you consider the new CPU a better overall design? Why?

ANSWER:

Out of the new CPU or the original CPU which one’s better depends on the programs that are run on the CPU. If the instructions of the programs contain atleast 30% ld/sd instructions then the original CPU will be considered a better design choice else the new CPU will be considered to be the better overall choice.

1. ***(a)*** Examine the difficulty of adding a proposed lwi.d rd, rs1, rs2 (“Load With Increment”) instruction to RISC-V. Interpretation: Reg[rd]=Mem[Reg[rs1]+Reg[rs2]]

ANSWER:

Load-store model for memory access uses RISC type ISAs. IT therefore means that only load/store instruction will be to access the memory. While on X86 most instructions will be allowed to operate directly on the data in the memory, MIPS or RISCV. Incrementing a 32-bit value at a particular memory address in MIPS/RISCV would require three types of instructions (load, increment, and store) to first load the value at a particular address into a register, increment it within the register, and store it back to the memory from the register. This form uses a register as an offset. An example could be when the codes will access an array where the index is computed at run-time.

The proposed LWI.d instruction would perform these tasks in one single instruction – load with increment where the content of register rs1 which holds a memory address is added to register rs2. Load operation will bring the content in memory to be brought into the registers rd. The suffix .d to lwi.d will respond to the shifter which will be used to scale the registers in rs2

***5.a1*** Which new functional blocks (if any) do we need for this instruction?

ANSWER:

ALU: adds the registers r1 and r2

SHIFTER: used to shift the offset in register r2

Therefore we don’t need a new hardware.

1. ***a2*** Which existing functional blocks (if any) require modification?

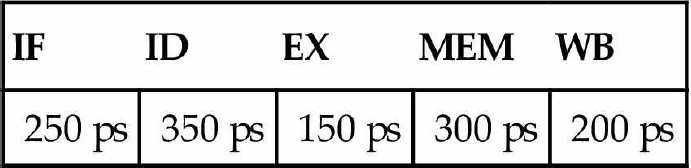
ANSWER:  
Control units require modification.

5.***a3*** Which new data paths (if any) do we need for this instruction?

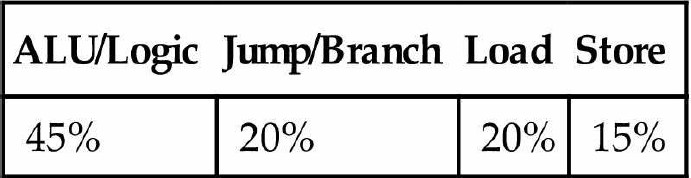
ANSWER:

New data paths are not needed

1. In this exercise, we examine how pipelining affects the clock cycle time of the processor. Problems in this exercise assume that individual stages of the datapath have the following latencies:



Also, assume that instructions executed by the processor are broken down as follows:



* 1. What is the clock cycle time in a pipelined and non-pipelined processor?

ANSWER:

For non-pipelined processor,

Clock cycle time = 1250 ps

For pipelined processor,

= 1250 / 5

= 250 ps

However, this is less than the highest latency in a single stage = 350 ps

Therefore, clock cycle time of pipelined processor = 350 ps.

* 1. What is the total latency of an ld instruction in a pipelined and non-pipelined processor?

ANSWER:

The latency will be same in both pipelined and non-pipelined processors.

The total latency of an ld instruction in a pipelined and non-pipelined = 1250ps.

* 1. If we can split one stage of the pipelined datapath into two new stages, each with half the latency of the original stage, which stage would you split and what is the new clock cycle time of the processor?

ANSWER:

The slowest stage is the instructions decode stage with latency of 350 ps.

We can split it into 2 stages each with latency of 175 ps.

Now the stage with highest latency is the Memory access stage with latency of 300ps which will be the new clock cycle time of the processors.

* 1. Assuming there are no stalls or hazards, what is the utilization of the data memory?

ANSWER:

Load and store instructions are the only instructions that access data memory. They will comprise 35% of the total number of instructions. The data memory will utilized 35% of the time.

* 1. Assuming there are no stalls or hazards, what is the utilization of the write-register port of

the “Registers” unit?

ANSWER:

ALU and load instructions are the only instruction that write back to write-register port of the “Registers” unit.

They comprise 65% of the total number of instructions.

The write-register port of the “Registers” unit is utilized 65% of the time.

1. What is the minimum number of cycles needed to completely execute n instructions on a CPU with a k stage pipeline? Justify your formula.

ANSWER:

For a k stage pipeline, the first instruction reaches the end of the pipeline in k cycles. After this, the remaining n-1 instructions execute at 1 cycle per instruction over the next n-1 cycles so, n instructions take k + n -1 cycles to execute in a k stage pipeline

1. **(a)** Assume that x11 is initialized to 11 and x12 is initialized to 22. Suppose you executed the code below on a version of the pipeline from Section 4.5 that does not handle data hazards (i.e., the programmer is responsible for addressing data hazards by inserting NOP instructions where necessary). What would the final values of registers x13 and x14 be?

addix11, x12, 5 addx13, x11, x12 addix14, x11, 15

ANSWER:

x13 = 33, x14 = 26 x11=22+5=27 x13=11+22=33 x14=11+15=26

1. Assume that x11 is initialized to 11 and x12 is initialized to 22. Suppose you executed the code below on a version of the pipeline from Section 4.5 *that does not handle data hazards* (i.e., the programmer is responsible for addressing data hazards by inserting NOP instructions where necessary).

What would the final values of register x15 be? Assume the register file is written at the beginning of the cycle and read at the end of a cycle. Therefore, an ID stage will return the results of a WB state occurring during the same cycle. See Section 4.7 and Figure 4.51 for details.

addix11, x12, 5 addx13, x11, x12 addix14, x11, 15 addx15, x11, x11

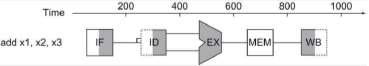
ANSWER:

x11=22+5=27 x13=11+22=33 x14=11+15=26 x15=27+27=54

1. Add NOP instructions to the code below so that it will run correctly on a pipeline that does not handle data hazards.

addix11, x12, 5 addx13, x11, x12 addix14, x11, 15 addx15, x13, x12

ANSWER:



|  |  |  |  |
| --- | --- | --- | --- |
| addix11, x12, | 5 | #x11 does not update to 27 (=22+5) until the add | |
|  |  | instruction below | |
| addx13, x11, | x12 | #x13 = 33 | |
| addix14, x11, | 15 | #x11 = | 26 |
| addx15, x11, | x11 | #x15 = | 54 |