Lab 1 Pipelining

Home assignment 1.1

How many cycles does it take before the pipeline is full?
Answer: 4 cycles

Express the CPI in terms of the number of “Elapsed Cycles” and “Hazards”
Answer: elapsed cycles / executed instructions

Home assignment 1.2

Study the data path model DP-1 (see Figure 1.1) and answer the following questions.

Find and write the name of all functional units that are not clocked by the system clock
Answer : REG file, ALU

How many 32-bits D-elements are there in DP-1?
Answer: 9

Home assignment 1.3

The program above consists of four instructions, each representing one of the four instruction classes shown in Table 1.1 on page 9.

Which instruction belongs to which instruction class?
Answer:
LW = LOAD instruction
ADD = ALU instruction
SW = STORE instruction
BEQZ = BRANCH instruction

What is the content (value) of the memory location (18) after the execution of the program above if (18) = 20_{16}
before the program execution
Answer:
40_{16}

Task 1.1

Which instruction was fetched?
Answer: LW

Give another clock pulse.
Which sub operation (according to Table 1.1) is
done by the instruction in the ID-stage?
Answer: RF - Register read

Give another clock pulse.

Where come the ALU operands from?
Answer: MX4, MX5 (R0 and constant)

What is the result from the ALU operation (study the output value from the ALU)?
Answer: 18_{16}

Why cannot the output value from the ALU be seen in the D-elements on the vertical clocking line to the MEM-stage?
Answer: Because it's a memory address, so the logic gate switches from normal constant mode to some fancy pants memory fetching mode and instead uses the ALU result as a address to fetch memory with instead of using directly as a constant.

Give another clock pulse.

Which address was generated (DMAR)?
Answer: 18_{16} was generated and returned 20_{16}

Thus, which of the sub operations in Table 1.1 was actually done in the EX-stage?
Answer: OB - Operand address calculation

Now, bring up the window showing the contents of the register file (click with the mouse on the register file). What is the contents of R2?
Answer: If you don't clock once more then 0_{16}

Give yet another clock pulse.
What is the contents of R2 now and why?
Answer: It is now 20_{16} because the CPU now did a writeback the the register

Instruction classes	IF	ID	EX	MEM	WB
LOAD instruction	IF	RF	OB	MR	RW
ALU instruction	IF	RF	AL		RW
STORE instruction	IF	RF	OB	MW	
BRANCH instruction	IF	RF	HB	RW/Nothing	
RESOURCES	MEM	REG	ALU	MEM	REG

IF - Instruction fetch

RF - Register read

AL - Arithmetic/logic operation

OB - Operand address calculation

HB - Jump target address calculation

MR - Memory read

MW - Memory write

RW - Register write

Task 1.2

We will study the sub operations that are done by the remaining instructions in the program. Give a clock pulse! The instruction ADD R1,R2,R2 will now be fetched to the processor. Clock the processor and answer the questions below.
Which sub operation according to Table 1.1 is done in the EX-stage?
Answer: AL

Which sub operation according to Table 1.1 is done in the MEM-stage?
Answer: Nothing

Before you clock the instruction to the WB-stage, you shall note the values of the registers R1 and R2. What is the content of R1 and R2?
Answer: R1 = 0, R2=10 000

Now, clock the instruction to the WB-stage.
What is the content of R1 now, and why?
Answer: R1 = 20 000

Now, fill in the sub operations for each pipeline stage in Table 1.1 that were done for the instruction class that the instruction belongs to.

Task 1.3

SW 18(R0),R1 is the next instruction to be executed. Clock the instruction through the data path and answer the following questions.

Which registers are read from the register file in the ID-stage?
Answer: R0, R1

Study the execution in the EX-stage. Which sub operation is done in the EX-stage?
Answer: OB

Which sub operation is done in the MEM-stage?
Answer: MW

The value of R1 is transported on a special bus. Why?
Answer: Because it will be written to memory instead of to a register

Which sub operation is done in the WB-stage?
Answer: Nothing

Fill in Table 1.1 the same way as before. When you shall determine in which stage a certain sub operation is done, it is irrelevant in which stage the corresponding functional unit is. The stage in which a sub operation is done is defined by the stage where the instruction is when the sub operation is performed. For example, a register write is done in the WB-stage even though the register file is draw in the ID-stage. This fact is important when you now shall identify the sub operations for BRANCH instructions.

Task 1.4

Finally, we will study which sub operations BEQZ R0,14 does.
Which sub operation is done in the ID-stage?
Answer: RF

Study carefully what is done in the EX-stage by looking at the multiplexers.
Which sub operation is done in the EX-stage?
Answer: AL

Note the result from the ALU.
Answer: 24

Which sub operation is done in the MEM-stage?
Answer: RW/Nothing

Now, study the state of the multiplexer MX1.
What does the multiplexer MX1 control?
Answer: Writing to the program counter

When you clock the instruction to the WB-stage, the value of program counter (PC) is changed. Which is the new value of the PC?
Answer: 24

Which sub operation was then actually done in the MEM-stage?
Answer: The mem write is redirected to the PC instead of reg/mem and writes the value from the ALU. I would consider that a sub operation, but apparently it is not.

For which instruction classes and pipeline stages are the sub operation ‘Nothing’ done?
Answer: ALU, STORE and BRANCH

Until now, we have executed each instruction at a time. An important performance measure is how many clock cycles it takes in average to ex ecute an instruction. The CPI is a measure of this.

Study the CPI at the bottom of the simulator window.
What is the CPI and why?
Answer: CPI is the average clock count per instruction. It is 5 because we have not enabled pipelining so each instruction will always take 5 clock cycles.

Home assignment 1.4

Write in Table 1.1 on page 9 in what pipeline stages the functional units Memory, ALU and Register file are used.

Why can the register file be used in two pipeline stages at the same
time without any structural conflict?
Answer: The ALU only reads and the WB only writes. If we write and read the same register bad things will happen, but if we do it to different registers it's fine.

Which requirements are posed on the memory in order to avoid structural conflict?
Answer: Reads and writes have to happen atomically

Home assignment 1.5

What do you expect the registers R1, R2, and R3 to contain after the execution of the program above? Write the answer in hexadecimal numbers.
Answer: R1 = 4, R2=8, R3=c

Task 1.5

How long is the execution time (counted in clock cycles)if you execute one instruction at a time
in the data path model DP-1?
Answer: 45 clock cycles (9 instructions * 5 cycles)

Now, active pipelining by clicking on this button option at the top of the simulator window!
Clock the processor until the first instruction is in the MEM-stage.
How many cycles does this take?
Answer: 4

Give yet another clock pulse! Which register is updated (written to) in this clock cycle and which registers are read?
Answer: R1 is both written to and read twice

Why can the newly written value directly be
read by an instruction in the ID-stage?
Answer: Since the WB write to the register is instant, there's no delay of the value even though they are on the same clock cycle.

How many instructions are executed concurrently (at the same time) in the pipeline?
Answer: 5

Now, clock the processor until the last instruction is in the WB-stage.
What is the content of R1, R2, and R3?
Answer: R1 = 2, R2 = 4, R3 = 3

Compare with your answer in Home assignment 1.5!
How many clock cycles did it
take to execute the program?
Answer: 45

If we neglect the number of clock cycles it takes to fill the pipeline (which you measured earlier), how much faster did the program execute when we turned on pipelining?
Answer: 13 clock cycles (9+4)
45/13 = ~3.5x faster

Task 1.6

You find the program above in the file exempel3.s. Load the program into the simulator and reset the program counter (press the button). Execute the whole program by clicking on the button.

What are the content of R1, R2, R3, and R4 after the execution of the program?
Answer: R1=1 , R2=0 , R3 =0 ,R4 = 2

Something has obviously went wrong! In order to under what went wrong, we will now execute the program once more. Therefore, reset the program counter. Clock the processor step-wise until ADD R2,R1,R1 is in the ID-stage.

Which value in R1 is read from the register file?
Answer: 0

Where in the pipeline (in what stage) in the correct value on R1?
Answer: EX

Clock the processor one step more so that ADD R3,R1,R1 is in the ID-stage.
Which value in R1 is read from the register file?
Answer: 0

Where in the pipeline (in what stage) is the correct value on R1?
Answer: MEM

Clock the processor one step more so that ADD R4,R1,R1 is in the ID-stage. Which value in R1 is read from the register file?
Answer: 1

Is this the correct value?
Answer: no

Clearly, you have found that the register file does not always contain the correct register values. You have in the exercise above found that you sometimes need to send/pass the result of one instruction to a succeeding instruction if the succeeding instruction needs the result from the preceding instruction. This requires that the pipeline keeps track of whether a destination operand in one instruction is used as source operand in the two succeeding instructions The mechanism that supports this is called Bypassing. We must be able to send register values back to the ID-stage from two places in the pipeline.
Which places?
Answer: EX, MEM

In the next data path model, Datapath Model 2 (DP-2), there is support for Bypassing.

Task 1.7

Bring up this new data path model by choosing DP-2 in the menu.
What are the differences between DP-2 and DP-1?
Answer: No functional units, only bypassing values from EX and MEM.

Clock the program until ADD R2,R1,R1 is in the ID-stage. Note the multiplexers' (MX2 and MX3) input data selection.

From where is R1 fetched?
Answer: From the EX stage

Now, clock the program one more step so the ADD R3,R1,R1 instruction is in the ID-stage. Again, note the multiplexers' (MX2 and MX3) input data selection. From where is R1 fetched?
Answer: The register

Clock the processor until the whole program has been executed and verify that the register values are as expected (see Home assignment 1.6)

Home assignment 1.7

What do you expect R1, R2, R3, and R4 to contain when the program above has executed, if we assume that (18) = 20_{16} before the execution?
Answer: R1=20, R2=20+20=40, R3=40+40=80, R4=80+80=160

Task 1.8

You find the program above in the file exempel4.s. Load the program into the simulator and reset the program counter. Execute the whole program by clicking on the button. What are the content of R1, R2, R3, and R4 after the execution of the program (compare to Home assignment 1.7)?

Something has obviously went wrong! In order to under what went wrong, we will now execute the program once more. Therefore, reset the program counter. Clock the program until the ADD R2,R1,R1 instruction is in the ID-stage. In which stage is the LW R1,18(R0) instruction?
Answer: EX

Clock the processor one more step.
Where in the pipeline (in what stage) is the correct value on R1?
Answer: In MEM

In which pipeline stage is the ADD R2,R1,R1 instruction?
Answer: EX

In which pipeline stage is the ADD R3,R1,R1 instruction?
Answer: ID

Why does the ADD R3,R1,R1 instruction get the correct register value?
Answer: Because at that clock the contents of 18(r0) has been fetched and passed by MEM

One instruction was executed erroneously.
Why did the problem occur at this instruction?
Answer: Because it needed one more clock to fetch the value of the LW instruction, and since the instruction after was dependent on that value it got a invalid value.

Task 1.9

Activate Delayed Load by clicking on the correct option at the top of the window, and verify that the options are as follows: Datapath Model 2
Clock the program until ADD R2,R1,R1 is in the ID-stage. In which pipeline stage is LW R1,18(R0)?
Answer: EX

Give a new clock pulse and study what happens. In which pipeline stage is ADD R2,R1,R1 now?
Answer:ID

In which pipeline stage is LW R1,18(R0) now?
Answer:MEM

The pipeline was halted (STALL) so all instruction the ID-stage and backwards stay where they are. The value that R1 shall be assigned is now read from memory. From where is R1 fetched (study MX2 and MX3)?
Answer: From MEM via MX2 and MX3

Execute the program to its end by clicking on the Run button and verify that the registers contain the correct values. How many clock cycles did it take to execute the program?
Answer:9

How many clock cycles should the program have taken to execute if the data dependence between the LOAD instruction and the first ADD instruction was not there?
Answer:8

Home assignment 1.8

Analyze the program above. What values do you expect in the registers R1, R2, R3, R4, and R5 after the execution of the program?
Answer: 0, 0, 9, 2d, 9, 2d

What values do R3, R4 and R5 have just before the ADD R3,R3,R2 instruction is executed?
Answer: 0, 0, 0

How many instructions will be executed?
Answer: 32 at this point, 35 in total

Task 1.10

You find the program above in the file exempel5.s. Load the program into the simulator and reset the program counter. Clock the processor until the branch instruction BNEZ R1,LOOP is in the ID-stage for the first time. Give yet another clock pulse.
Which instruction was fetched?
Answer: add r3,r3,r2

Give yet another clock pulse. Which instruction was fetched now?
Answer: same

What is the value of the program counter?
Answer: 20

Give yet another clock pulse. Which instruction was fetched now and what is the value of the program counter?
Answer: addi r2, r2, #1 and PC 14

It seems like some instructions have been fetched into the pipeline that should not have been there. Give yet another clock pulse. What is the value in R3 and why is this value wrong (compare to Home assignment 1.8)?
Answer: 0, should be 9. Because it doesn't stall when it jumps so the instructions run will be wrong because they're not the ones we jumped to.

How many clock cycles does it take from when the branch instruction is in the ID-stage (and thus is decoded) until the program counter is loaded with the branch target address?
Answer: 3

Task 1.11

Deactivate “Delayed Branch” by clicking on the corresponding button at the top of the simulator window. Verify that you have the following options: Datapath Model 2
Execute the program again by clocking the processor until the branch instruction BNEZ R1,LOOP reach the ID-stage for first time. Give yet another clock pulse.
Which instruction was fetched?
Answer: None, so it’s still ADD R3, R3, R2

Apparently, the processor does not fetch any new instructions. Count the number of clock cycles until the processor starts to fetch instructions again.
How many clock cycles does it take until the processor starts to fetch instructions again?
Answer: 2 more, 3 from the ID

Execute the program to the end. Verify that the registers now have the correct values by comparing the register contents to what you came up with in Home assignment 1.8.
How many clock cycles did it take to execute the whole program?
Answer: 63

Compare this number to the number of instructions you came up with in Home assignment 1.8. Explain the difference!
Answer: We expected 35, it was 65. It needed to stall a lot because it did a lot of branches/jumps

Task 1.12

Bring up the next data path model by choosing DLX in the menu. Study how the branch target address calculation now can be done two cycles earlier when an extra adder (ADD) has been introduced in the ID-stage. Also note how the test of the branch condition has been moved to the ID-stage. Load the same program as before (exempel5.s)
Clock the processor until the branch instruction BNEZ R1,LOOP is in the ID-stage for the first time. Give another clock pulse.

Which instruction was fetched now and why?
Answer: addi R2, R2, #1, because there was an extra adder at the ID which adds a value to the PC if the branch instruction is true.

Execute the program to the end.
How many clock cycles did it now take to execute the whole program?
Answer: 47 clock cycles

How much faster did the program execute now as compared to with DP-2?
Answer: 63/47 = ~40% faster

Note the CPI for the program. Comments?
Answer: It doesn’t seem like the CPI accounts for stalls, so the CPI is lower on DP-2 than DLX even though DLX is faster.

Task 1.13

The program above is found in the file exempel6.s. Load the program into the simulator and reset the program counter. Execute the program to the end. What is the resulting CPI?
Answer: 2

The CPI (average number of Cycles Per Instruction) for a given program is a measure of how good a pipeline implementation is on eliminating hazards and pipeline stalls. For the program loop, which is executed 50 times, an approximate value of the CPI the quote between the number of cycles it takes to execute one iteration and the number of instructions in each iteration (NOP is not counted as an instruction). This is a good approximation since the number of cycles to fill the pipeline (4 cycles) is negligible compared to the number of in structions that are executed in the program loop (200 instructions in the program above).
Use the approximation above and explain which hazards that occur in each iteration of the loop.
Answer: There are multiple hazards in each loop iteration. One is that add uses a register that LW sets and the other one is that before the branch we change the value of r2 which is required by bnez to decide if it will jump or not. So approx 2 hazards per loop iteration.

By using the number of detected hazards (Hazard) and the fact that the program executes the loop 50 times, identify how many clock cycle delays that occur each iteration.
Answer: 2 stalls, which is 3+4=7 clock cycles (3 for lw, 4 for branch)

Task 1.14

The program above is found in the file exempel7.s. Load the program into the simulator and reset the program counter. Execute the program to the end. What is the resulting CPI?
Answer: 1.26

Compare the CPI with the proportion of branch instruction that are executed. Comments?
Answer: There's 4 instructions per loop with one stall per loop, which makes 25% of the cycles stalled which makes the CPI to be ~1.25

What speed-up was achieved as compared to Task 1.13?
Answer: Since the LW instructions dependency was put one place later it didn't need to stall, so there was only one hazard per loop instead of two.

Task 1.15

Activate the option and load the program above, which you find in the file exempel8.s, and reset the program counter. Execute the program to the end.
What is the resulting CPI this time?
Answer: 1.34

Task 1.16

Calculate the CPI for the program above without executing the program!
Answer: (1+(49*4)+1)/(1+(49*3)+1) = 1.3288590604026846