

0x00c6ba23(16) = 0000 0000 1100 0110 1011 1010 0010 0011

Immediate[11:5]: 0000000

Rs2: 01100

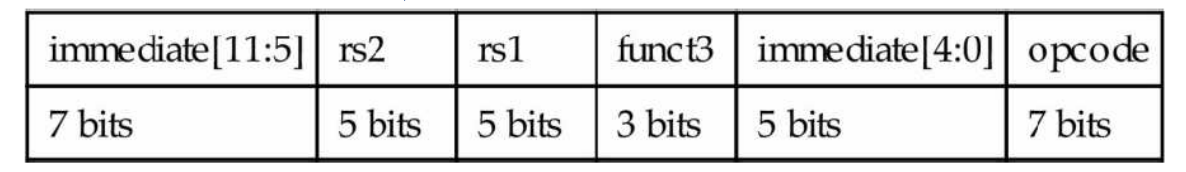
Rs1: 01101

Funct3: 011

Immediate[4:0]: 10100

Opcode: 0100011

According to the opcode: 0100011, and the func3 fields: 011, we can see that this instruction is ‘sd’ of ‘S’ format and the instruction is as x12, 20(x13). The ‘S’ format instruction as shown in the following table.



1.1

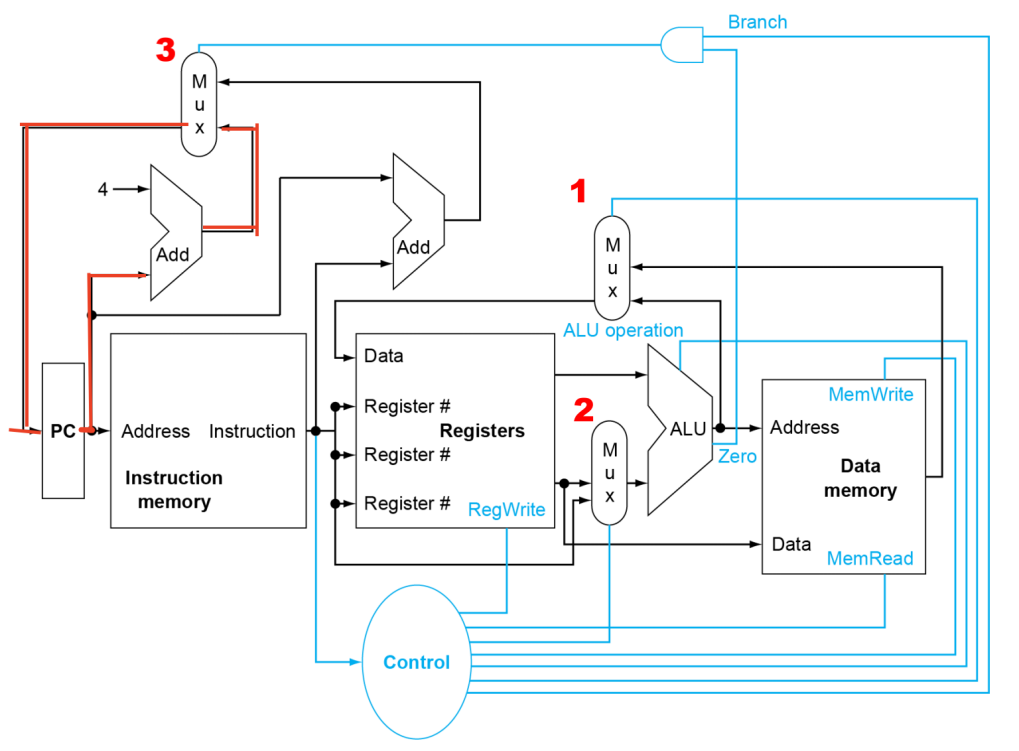
According to the table shown below, the ALUop bits for ‘sd’ instruction are 00.

The ALU Control lines inputs the value 0010.

1.2

For there is no branch instruction, the new PC address should be the next PC address, which is PC + 4. And the path through is highlighted in red color in the picture

below:



1.3

There are three mux units as shown in the picture above: ALUsrc, PCsrc and MemtoReg.

ALUsrc is 1 for ‘sd’ instruction, and put the extended immediate bits into rs1(x13), then store the content of memory in rs2(x12). i.e. The input1 is rs1, and the input2 and output is 0x00..0014.

PCsrc is 0 for there is no branch instruction. Input: PC, 4; Output: PC + 4.

MemtoReg is irrelevant. Input: rs1+0x00..0014.

1.4

As shown in the picture in 1.2, ALU inputs are rs1 (Reg[x13]) and 0x00..014 from the mux controlled by ALUsrc.

Branch adder inputs: PC and 0x0..014 shifted by 1 (0x0..28).

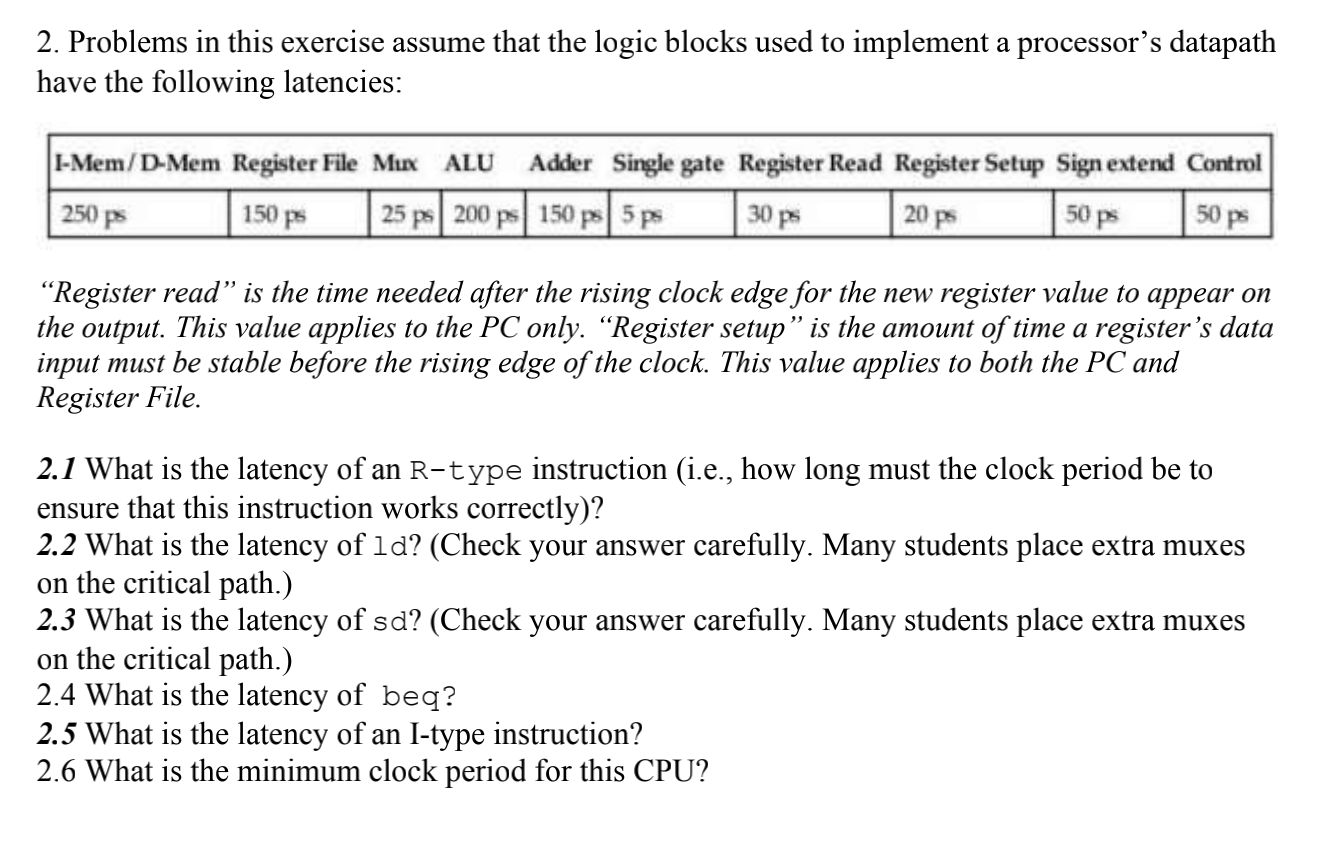
PC adder inputs: PC and 4.

1.5

Rs1 has 5-bit instruction field[19-15], which is x13.

Rs2 has 5-bit instruction field[24-20], which is x12.

For the Reg write control signal is disabled for the sd instruction, the register input port has irrelevant bits.



2.1

R-Type Instruction:

1. PC Register Read delay following rising edge of clock: 30 ps

2. Instruction Memory: 250ps

3. Read Register File: 150 ps

4. Mux to pick data from either R2 or sign extended bit data from immediate

field of instruction = 25 ps

[note - there is no number given for sign extension delay, so we can assume

it is less than the register file delay of 150 ps]

5. ALU delay: 200 ps

6. Mux for WB to Register File of result from ALU: 25 ps

7. Write back setup time for Register File registers: 20 ps

So the total latency is 700ps.

2.2

For ld instruction:

1. PC Register Read delay following rising edge of clock: 30 ps

2. Instruction Memory: 250ps

3. Read Register File: 150 ps

4. Mux to pick data from either R2 or sign extended bit data from immediate field of

instruction = 25 ps

[note - number given for sign extension delay of 50 ps is less than the 150 ps for register

file read access, so we can assume that mux control signal is designed to fire only when

both inputs are ready]

5. ALU delay: 200 ps

6. Read data from Data Mem, whose address provided by ALU: 250 ps

7. Mux for WB to Register File of result from Data Mem: 25 ps

8. Write back setup time for Register File registers: 20 ps

So the total latency is 950ps.

2.3

For sd instruction:

1. PC Register Read delay following rising edge of clock: 30 ps

2. Instruction Memory: 250ps

3. Read Register File: 150 ps

4. Mux to pick data from either R2 or sign extended bit data from immediate field of

instruction = 25 ps

[note - there is no number given for sign extension delay, so we can assume it is less than

the register file delay of 150 ps]

5. ALU delay: 200 ps

6. Write Data from R2 to Data Mem: 250 ps

So the total latency is 905ps.

2.4

For beq instruction:

1. PC Register Read delay following rising edge of clock: 30 ps

2. Instruction Memory: 250ps

3. Read Register File: 150 ps

4. Mux to pick data from either R2 or sign extended bit data from immediate field of

instruction = 25 ps

[note - there is no number given for sign extension delay, so we can assume it is less than

the register file delay of 150 ps]

5. ALU delay: 200 ps

6. single gate delay AND of ‘zero’ output from ALU and ‘Branch’ control output from

Control unit: 5ps

7. Mux for Branch address to PC: 25 ps

8. Write back setup time for PC register: 20 ps

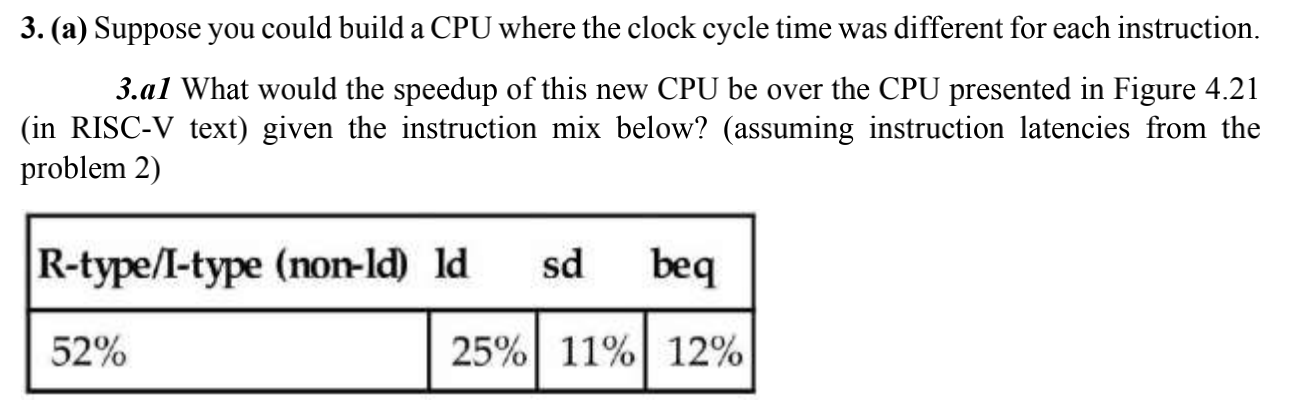
So the total latency is 705ps.

2.5

The latency of I-type instruction is same as the R type instruction, which is 700ps.

2.6

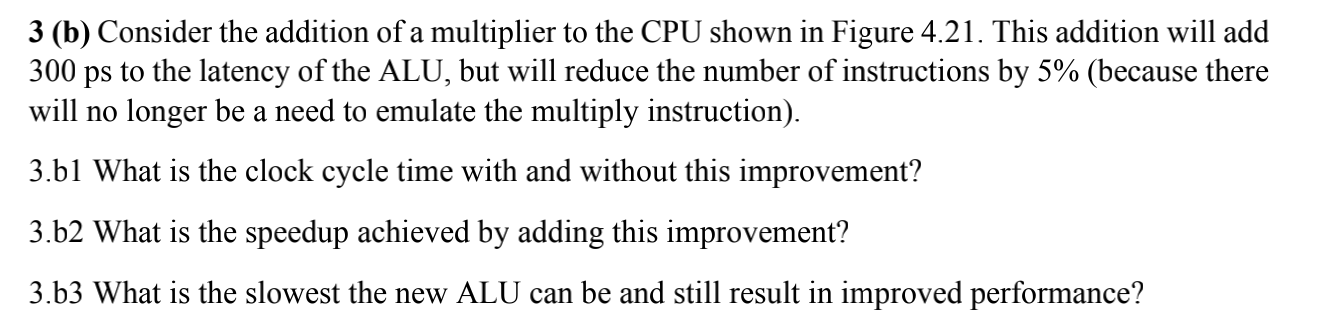
The minimum clock period for this CPU should be the longest latency of instructions, which is 950 ps.



New CPU clock time:

0.52\*700ps + 0.25\*950ps + 0.11\*905ps + 0.12\*705ps = 785.6ps

Speedup = 950ps/785.6ps = 1.21



1. b1

Without the improvement: clock cycle time = 950ps

With the multiplier improvement: clock cycle time = 950+300 =1250 ps

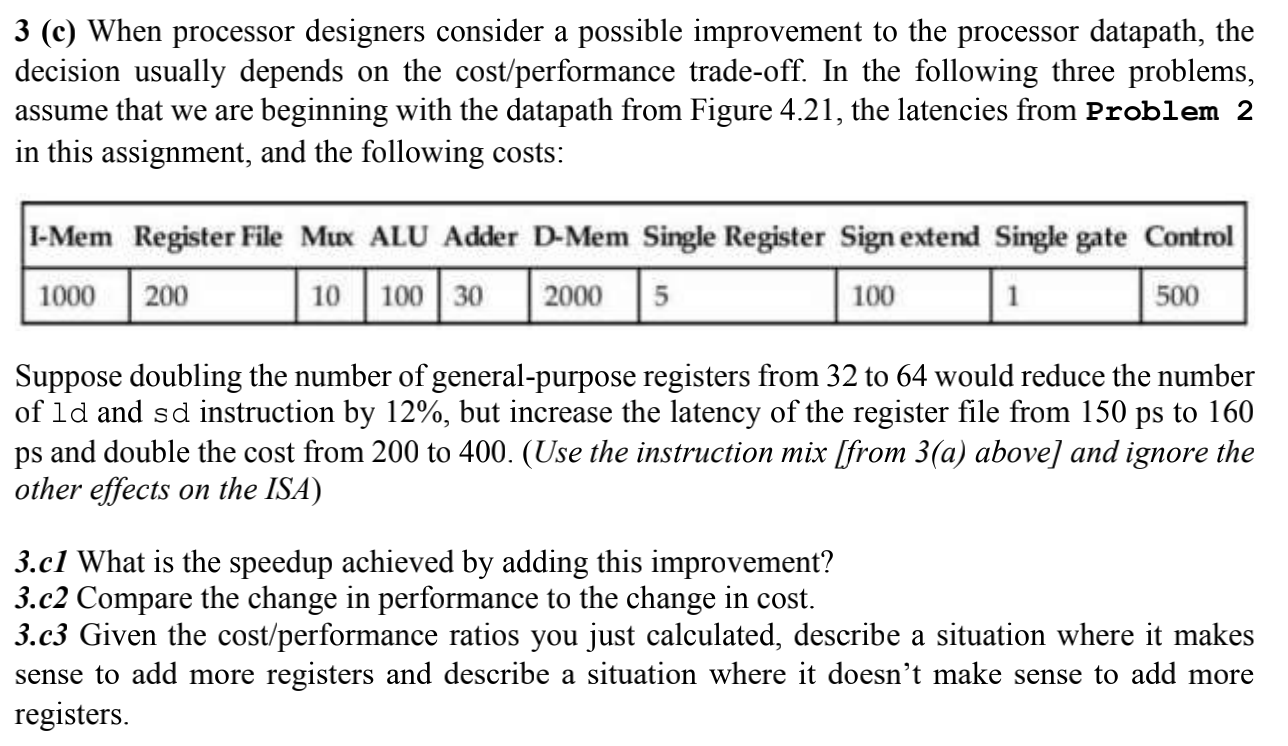
1. b2

Speedup = 950ps/(1250ps\*0.95) = 0.8

1. b3

1250ps\*0.95 = 1187.5ps which is larger than 950ps

To improve the performance, 950/[(950 + x) \* 0.95]should larger than 1, then x < 50ps. Therefore the slowest new ALU can be 200 + 50 = 250 ps

3.c1

Speedup = 950ps/(950\*(1-(0.25+0.11)\*0.12)) = 1.045,

which is improve by 4.5%

1. c2

The change in performance is improved by 4.5% as in 3.c1. However the change in cost is :

Original cost = 1000 + 200 + 10\*3 + 100 + 30\*2 + 2000 + 5 + 100 +500 + 1 = 3996

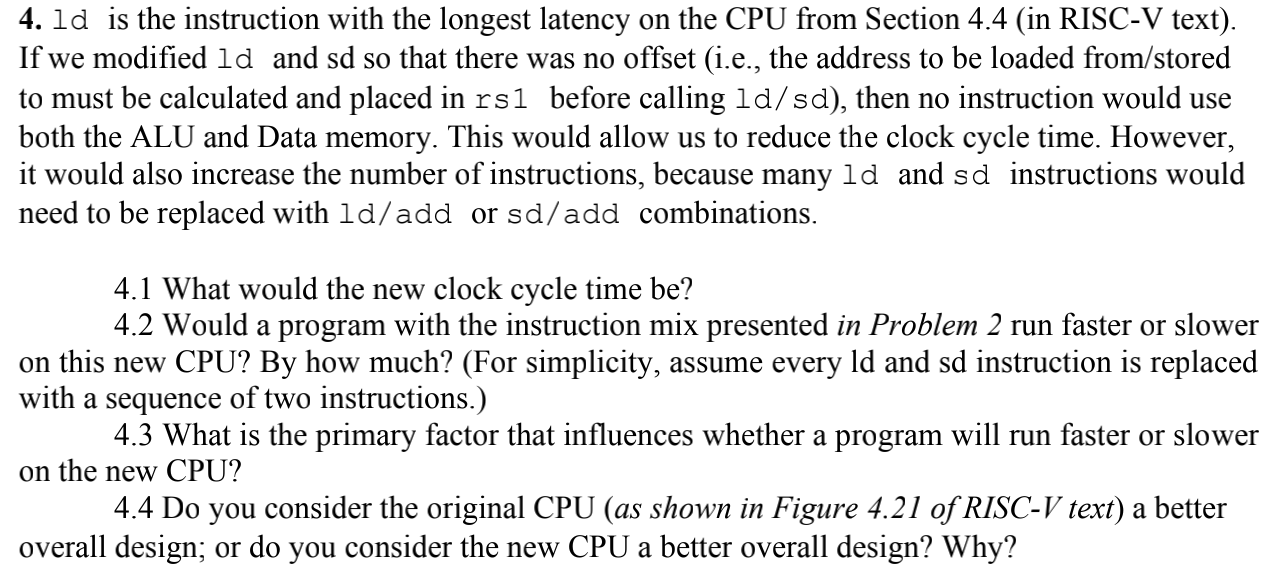
New cost = 1000 + 400 + 10\*3 + 100 + 30\*2 + 2000 + 5 + 100 +500 + 1 = 3996

The cost increase 4196/3996 -1 = 5%.

1. c3

cost/performance ratios = 5%/4.5% = 1.25

As can be seen from the above, the speed has increased by 4.5% and the price has increased by 5%. This is not a big contrast, so it can be improved when the demand for CPU speed is high. If it's just common use, when it doesn't need high processing speed, it doesn't need this kind of improvement.



4.1

After the modification, ld instruction has no step 4 and 5 in 2.2. i.e. no mux delay and ALU delay.

Therefore the new clock time for ld is 950 - 200 - 25 = 725ps

Sd instruction has no mux delay and ALU delay.

The new clock time for sd is 905 - 200 - 25 = 680ps

4.2

The ld and sd instruction are 36% of all instructions. Therefore the new number of instructions would be 1.36 times larger than old number. And the new cycle time would be 725\*1.36 = 986ps, which is slower(950ps before).

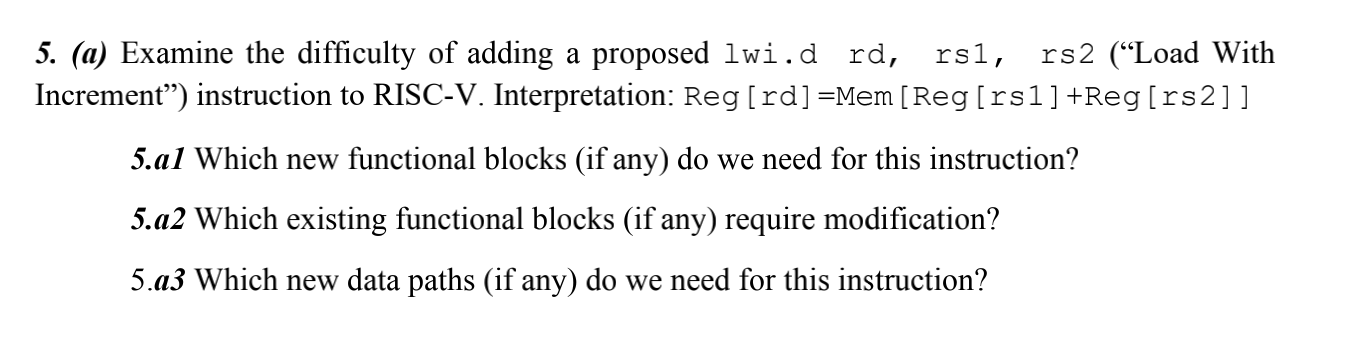
Speedup = 950/986 = 96.3%

4.3

The primary factor is the ratio of ld and sd instructions to the total number of instructions. For the cycle time is decreased for every instruction, but only ld and sd instructions are doubled.

4.4

I think the original CPU is a better overall design, because according to 4.2, the new CPU runs slower by 4% than the original. Unless we decrease the percentage of ld and sd instructions. When the number of these two kinds of instructions is less than 31% of total(725\*1.31 = 949.75 < 950), the new CPU would be better.



5.1

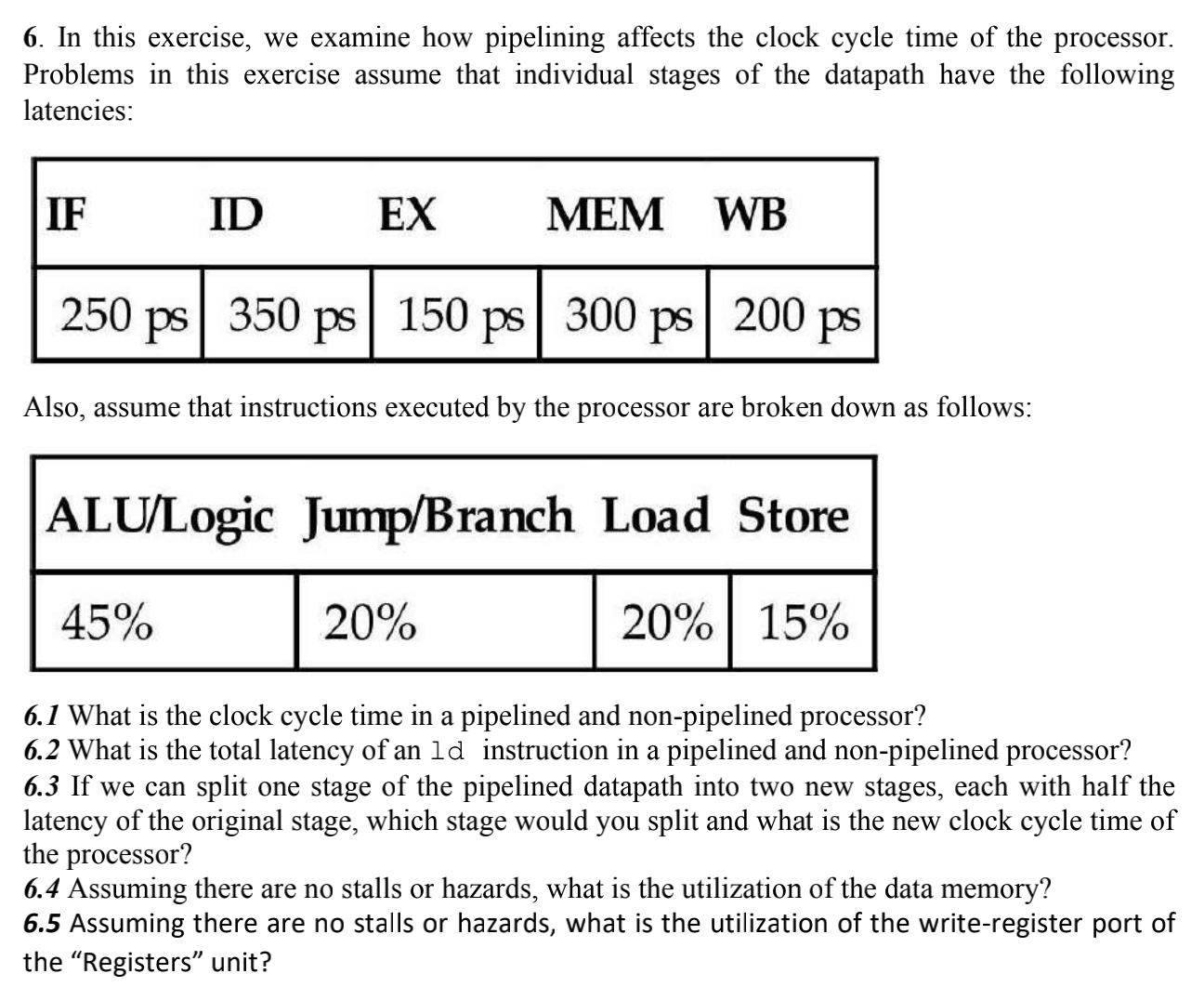
No new hardware is necessary. The ALU could be used to add the 2 registers rs1 and rs2, the shifter could be used to shift the offset in register rs2.

5.2

Control unit needs modification.

5.3

No new data paths are needed.



6.1

Pipeline: 350ps(the stage with largest latency)

Non-pipeline: 250 + 350 + 150 + 300 + 200 = 1250ps

6.2

Both pipelined and non-pipelined processor’s latency are 1250ps.

6.3

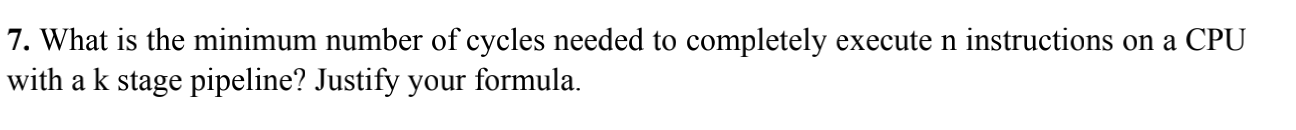
I would split ID stage for it is the slowest one. Now the slowest stage is MEM stage which is 300ps, therefore the new clock cycle time is 300ps.

6.4

For load and store need to use the data memory, the utilization is 20% + 15% = 35%

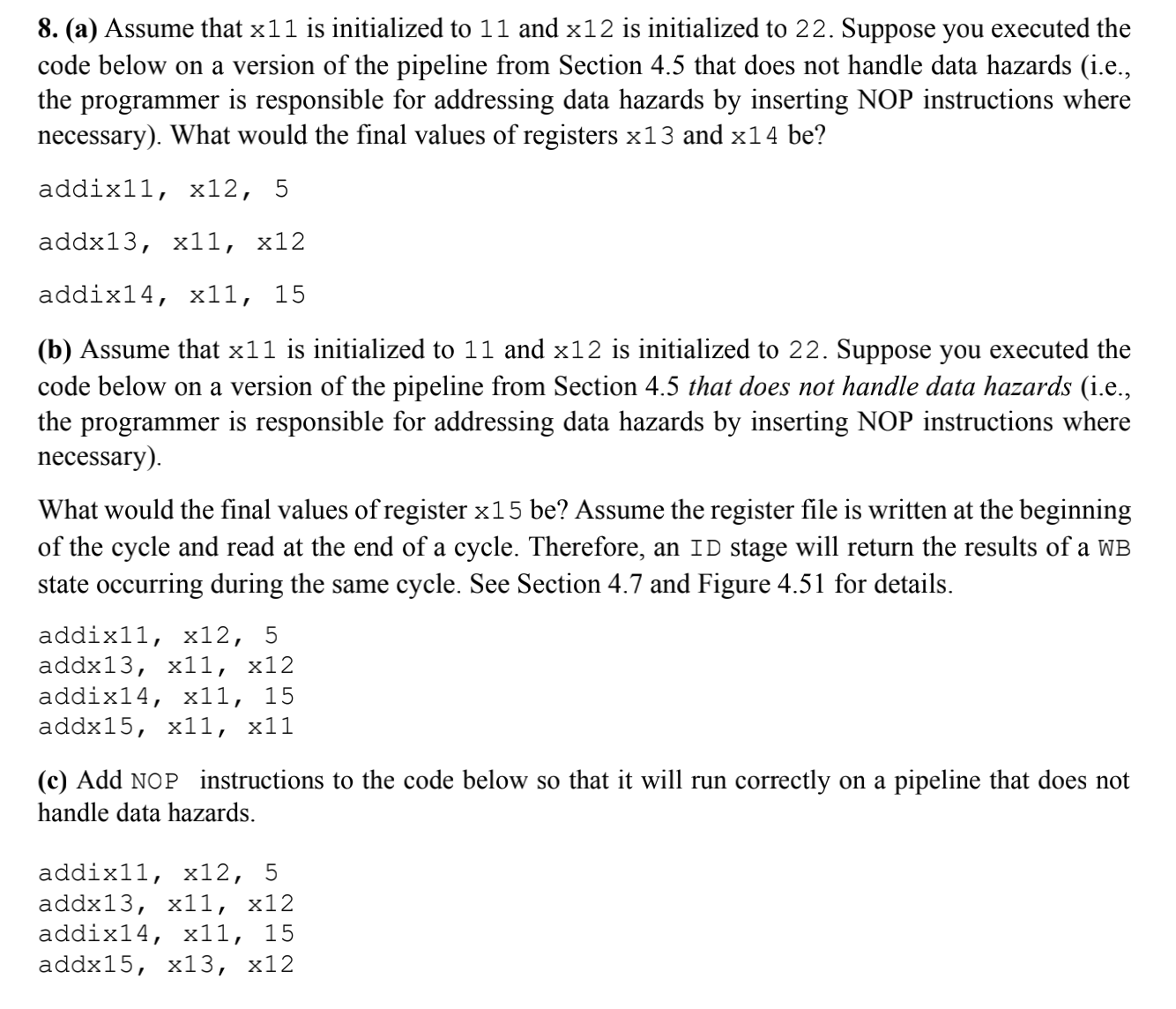
6.5

For ALU and load would write back to write-register port, the utilization is 45% + 20% = 65%.



k+n-1

For the first instruction, it needs k cycles, and each of the next n-1 instructions only need 1 cycle in pipeline. Therefore the number of total cycle needed is k + n -1

(a)

addix11, x12,5 #x11 does not update to 27 until the fourth line

addx13,x11,x12 # x13 = 11 + 22 = 33

addix14,x11,15 #x14 = 11 + 15 = 26

x13 = 33

x14 = 26

(b)

addix11, x12, 5

addx13, x11, x12 #x11 = 11

addix14, x11, 15 #x11 = 11

addx15, x11, x11 #x11 is updated to 22 + 5 = 27

X15 = 54

(c)

addi x11, x12, 5

NOP

NOP

add x13, x11,x12 (x11 data hazard)

addi x14, x11,15

NOP

Addi x15,x13,x12