ma1478
N17006432

NYU | TANDON SCHOOL OF ENGINEERING

Examination Book

A student using or receiving unauthorized assistance during an examination, as from notes or other students, is in violation of academic regulations and is subject to academic discipline, including forfeiture of credit for the course, probation and dismissal from NYU Tandon School of Engineering.

Name MAHASMRTI ADHURI

Subject CSA I.D. No. ma1478

Instructor AZEEZ BHAVNAGARWALA

Exam. Seat No. _____ Section B

Date 11/08/22 Grade _____

NEW YORK UNIVERSITY

1. (i) if (g > h)

```
    g = g + 1;
else
    h = h - 1;
```

RISC V code:

```
    ELSE  // (go to ELSE if (g <= h))
    ble  x5, x6
    addi  x5, x5, 1    // g = g + 1;
    beq  x0, x0, EXIT  // Unconditional EXIT
ELSE : addi  x6, x6, -1    // h = h - 1;
EXIT :
```

(iii) if (g<=h)

g=0;

else

h=0;

RISCV code:

```
    ble  x5, x6, IF   // go to IF label when (g<=h)
    addi x6, x6, 0    // h=0;
    beq  x0, x0, EXIT // Unconditional EXIT
IF: addi x5, x0, 0    // g=0;
EXIT:
```

(or)

```
    blt  x6, x5 ELSE  // goto ELSE if h<g
    ADDi x5, x0, 0    // g=0
    BEQ  x0, x0, EXIT // Unconditional EXIT
ELSE: ADDi x6, x0, 0  // h=0
EXIT:
```

(2)

```
Add  x5 , x5 , x6 ;
sub  x6, x5, x6 ;
sub  x5, x5 , x6 ;
```

3)

(3.1)

I- mem is read; two registers are read, and a register is written.

140 pJ + 70*2 pJ + 60j = 340pJ

Therefore, the energy spent to execute an addi instruction in a single cycle design and in the five stage pipelined design is 340pJ

(3.2)

140 pJ + 2*70pJ + 60pJ + 140pJ

= 480 pJ

(3.3) I- mem + 2 registers

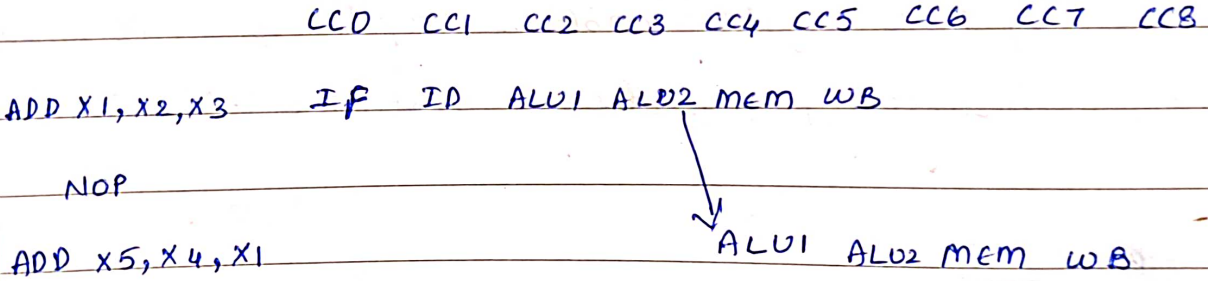
= 140pJ + 2*70pJ

= 280pJ.

(4) (4.1)

Data forwarding alone does not suffice here because by the time the first ALU instruction completes the second ALU (execute) step, the second ALU instruction is already in the first ALU step & thus it's too late. Therefore, a NOP between the 2 instructions, as well as data forwarding between the second ALU step & the decode step.

(4.2)



(5) addi s0, zero, 5 # result = 5 Runs 1

L1:

```
bge zero, s0, Done
addi s0, s0, -1
jL1
```

} The loop runs for 6 iterations.

DONE:

Total no. of instructions = 1 + 3 × 6 + 1

= 20 instructions.

For 5-Stage pipeline,

No. of cycles = 5 + (20 - 1) = 24 cycles.

CPI = 24/20 = 1.2 //

(b)

XOR s1, s2, s3	IF	ID	EX	ME	WB				
addi s0, s3, -4		IF	ID	EX	ME	WB			
lw s3, 16(s7)			IF	ID	EX	ME	WB		
sw s4, 20(s1)				IF	ID	EX	ME	WB	
or t2, s0, s1					IF	ID	EX	ME	WB

↳ 5th cycle.

In the 5th cycle,

→ Result of XOR is written back to register s1 in WB.

→ Data present in register s4 is read by the sw instruction in ID.

(7) (7.1)

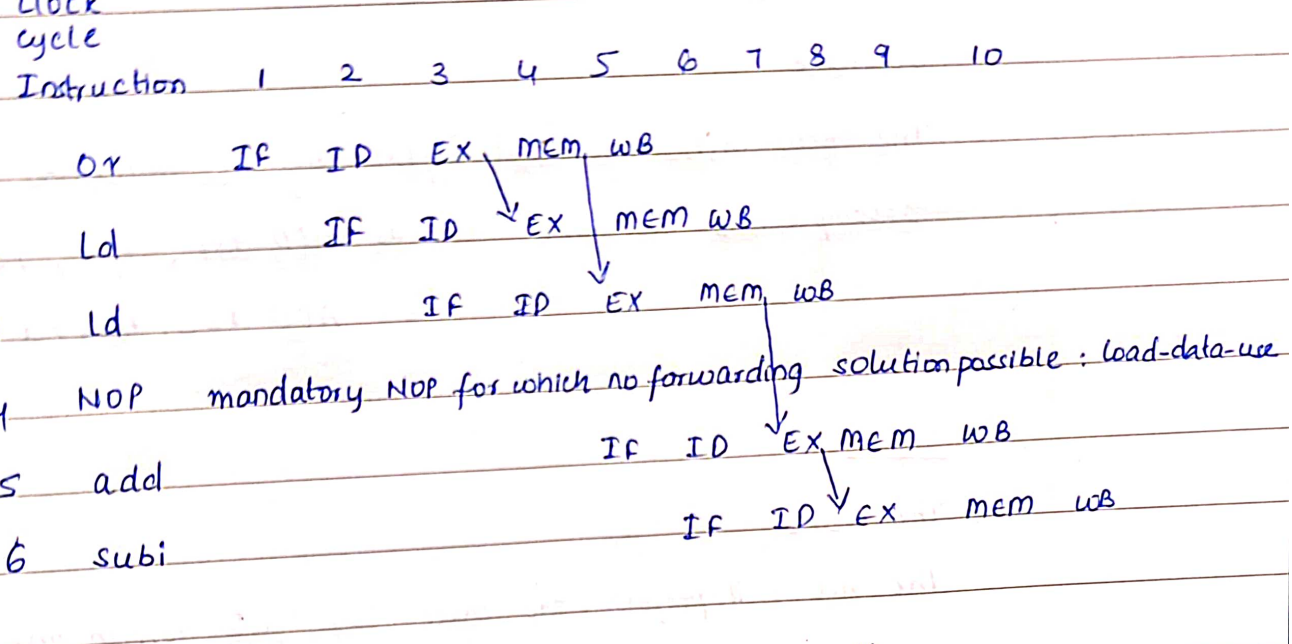
Hazards identified:

```
Or   X13, X12, X11
ld   X10, 0(X13)      Ex to 1st RAW Hazard
ld   X11, 8(X13)      Ex to 2nd RAW Hazard
add  X12, X10, X11    MEM to 1st RAW [load-use-data] & MEM to
                      2nd Hazards.
Subi X13, X12, 16     Ex to 1st RAW Hazard.
```

NOPs introduced to resolve Hazards:

```
Or   X13, X12, X11
NOPs
NOPs
ld   X10, 0(X13)      Ex to 1st RAW Hazard resolution with 2 NOPs
ld   X11, 8(X13)      Ex to 2nd RAW Hazard resolved as well from
                      above 2 NOPs.
NOPs
NOPs
add  X12, X10, X11    MEM to 1st RAW [load-use-data] & MEM
                      to 2nd Hazards resolved with 2 NOPs
NOPs
NOPs
subi X13, X12, 16     Ex to 1st only RAW Hazard resolved with
                      2 NOPs
```

7.2



(1) A = X B = X (no instruction in EX stage yet)

(2) A = X B = X (no instruction in EX stage yet)

(3) A = 0 B = 0 (both operands of the or instruction: X11, X12 come from Reg file).

(4) A = 2 B = 0 (base (RS1) in first ld (X13) taken from EX/MEM of previous instruction)

(5) A = 1 B = 0 (base (RS1) in 2nd ld (X13) taken from MEM/WB of a previous instruction)

(6) A = X B = X (no instruction in ex stage yet because NOP introduced to resolve mem to 1st).

(7) A = 0 B = 1 (RS2 in the add instruction is X11 which is forwarded from MEM/WB of 2nd ld, the result of 1st ld (X10) has already been written into Reg File in CC 6 no forwarding necessary for first operand.

(8) A = 1 B = 0 (RS1 of subi instruction forwarded from EX/MEM of add,

(8)

(A) Instructions / program

Decrease

The new instructions will replace any two instruction sequence that accomplished the same such as ADD rs1, rs2, rd

ADD rs3, rd, rd.

CPI

The same

The ALU will perform the three-way addition in one cycle. Also acceptable increase because more RAW

Circuit complexity

Increases

Three-operand ALU is more complex than a two operand one.

(B) Instructions / program

The same

No difference to instructions.

CPI

Increases.

All instructions now use the same ALU. ALU operations now

have to stall.

circuit complexity

Decreases.

Now there is one less adder/ ALU cycle.

(C) Instructions / program

The same or decrease

If more registers enable the compiler to avoid loads and stores, instructions / program decreases. Otherwise it remains the same.

- CPI

The same.

Does not affect the actions of each instructions.

circuit complexity.

Increases.

More registers complicate the register file.