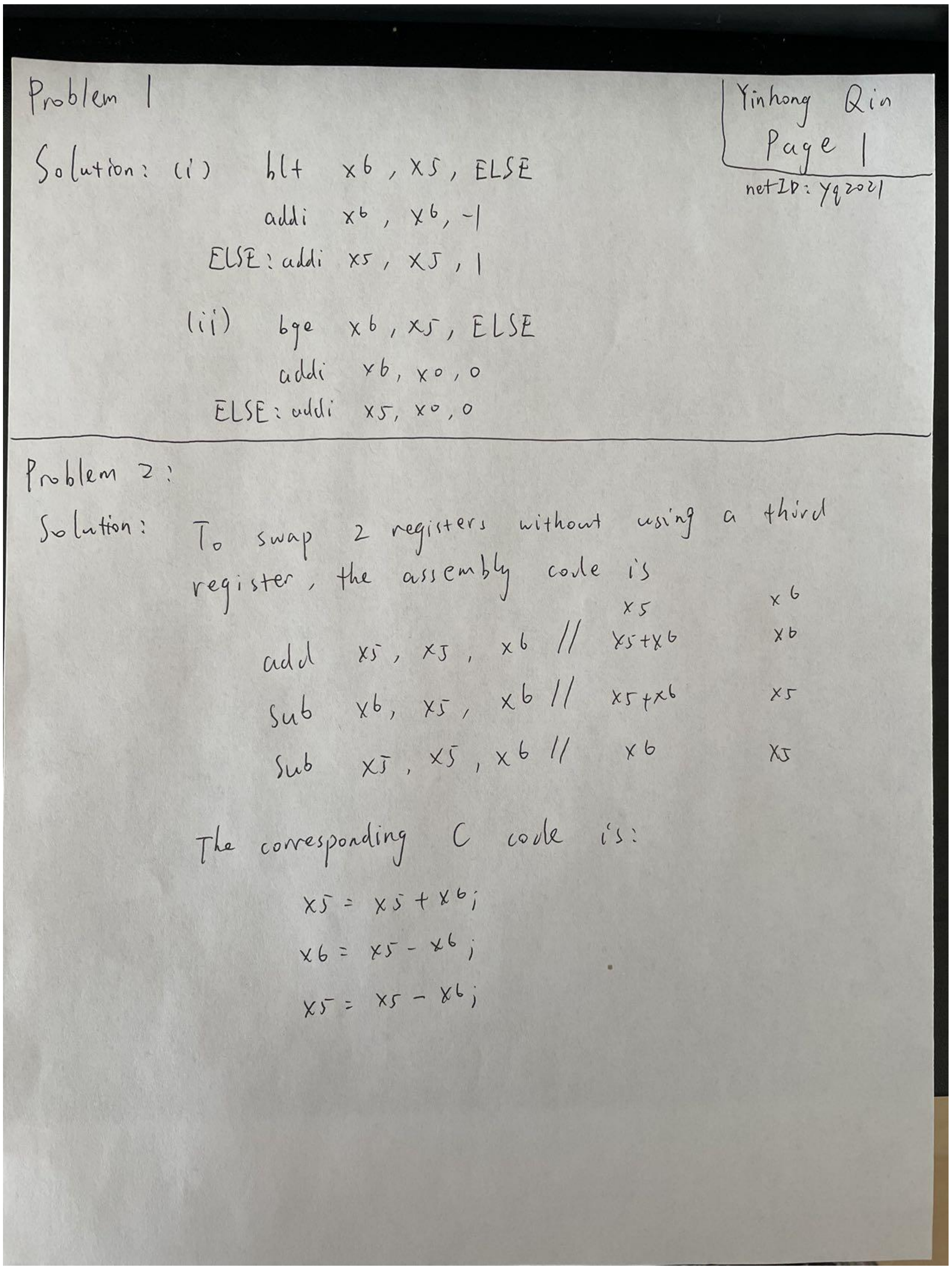


Yinhong Qin
Page 1
netID: yq2021

Problem 1

Solution: (i)

```
      blt  x6 , x5, ELSE
      addi x6 , x6, -1
ELSE: addi x5 , x5 , 1
```

(ii)

```
      bge  x6 , x5 , ELSE
      addi x6, x0 , 0
ELSE: addi x5, x0 , 0
```

Problem 2:

Solution: To swap 2 registers without using a third register, the assembly code is

```
                        x5         x6
add  x5 , x5 , x6 //  x5+x6        x6
sub  x6, x5 , x6 //   x5+x6        x5
sub  x5 , x5 , x6 //    x6         x5
```

The corresponding C code is:

```
x5 = x5 + x6;
x6 = x5 - x6;
x5 = x5 - x6;
```

Problem 3:

Yinhong Qin
yq2021
Page 2

Solutions:

3.1 "and" is a R-Type instruction:

The energy consumption for single-cycle design:

Access Instruction memory: 140 pJ

Read 2 registers: 70 pJ x2 = 140 pJ

Write back to registers: 60 pJ

140 + 140 + 60 = 340 pJ

Thus, the total energy consumption for a single-cycle design "and" instruction is 340 pJ.

For a five-stage pipelined design:

3.2 "sw" is a S-Type instruction:

The energy consumptions are:

Access Instruction memory: 140pJ

Read 2 registers: 70 pJ x2 = 140pJ

Write data memory: 120 pJ

140 + 140 + 120 = 400 pJ

Thus the total energy required is 400pJ

3.3 "beq" is a "SB-Type" instruction:

The total energy consumption is:

Access Instruction memory: 140pJ

Read 2 registers: 2 × 70pJ = 140pJ

There is no Register write or Data memory write.

Thus the total energy required is 140+140 = 280pJ

Problem 4:

Solution:

4.1 & 4.2

We need to use both bubbles and data forwarding here to resolve the data hazard here.

Original pipeline (we use EX_1 and EX_2 to express 2-cycles ALU)

add x1, x2, x3	IF	ID	EX_1	EX-2	MEM	WB		
add x5, x4, x1		IF	ID		EX_1	EX_2	MEM	WB

If we only use data forwarding, once the first "EX_2" give the result, the "EX_1" of 2nd instruction is done which is too late. Thus we also need a bubble / NOP between these 2 instructions.

Thus, we can have

	CC0	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8
add x1, x2, x3	IF	ID	EX-1	EX_2	MEM	WB			
NoP			- -	- -	forwarding - -				
add x5, x4, x1			IF	ID	EX_1	EX_2	MEM	WB	

Yinhong Qin
yq 2021

Problem 5: Solution:

Assume the multicycle processor has data forwarding unit and data hazard detection unit.

The first few cycles are:

addi s0, zero, 5	IF	ID	EX	mem	WB							
L1: bge zero, s0, Done		IF	ID	EX	mem	WB						
addi s0, s0, -1			IF	ID	EX	MEM	WB					
J L1				IF	ID	EX	MEM	WB				
bge zero, s0, Done					IF	ID	EX	mem	WB			
addi s0, s0, -1						IF	ID	EX	mem	WB		
J L1							IF	ID	EX	mem	WB	
bge zero, s0, Done								IF	ID	EX	mem	wb

Thus, the total cycles this program needed is 17

the total number of instructions is: 16

Thus, the CPI is: $\frac{17}{16} = 1.0625$

Problem 6

	Cycle1	Cycle2	Cycle3	Cycle4	Cycle5			
xor s1, s2, s3	IF	ID	EX	MEM	WB			
addi s0, s3, -4		IF	ID	EX	MEM	WB		
lw s3, 16(s7)			IF	ID	EX	MEM	WB	
sw s4, 20(s1)				IF	ID	EX	MEM WB	
or t2, s0, s1					IF	ID	EX	mem wb

Problem 6 (continued)

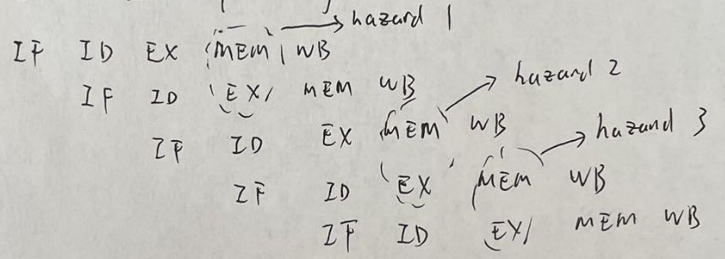
i. In the fifth cycle,
The instruction memory, register S7 are being read.
The register S3 is being written

Problem 7:

Solution:

7.1 If there is no data forwarding and hazard detection

or x13, x12, x11
ld x10, 0(x13)
ld x11, 8(13)
add x12, x10, x11
subi x13, x12, 16



Hazard 1: when "ld" access x13, its value hasn't been updated.
To resolve this, add 2 NOPs between "or" and "ld"

Hazard 2: when "add" accesses x11, its value hasn't been updated.
To resolve this, add 2 NOPs between "ld" and "add"

Hazard 3: when "subi" access x12, its value hasn't been updated
To resolve this, add 2 NOPs between "add" and "subi"

7.2 Solution:

Yinhong Qin
yq2021

	~~Cycle 1~~	~~Cy~~
~~or x13, x12, x11~~		
~~ld x10, 0(x13)~~		
~~ld x11, 8(x13)~~		
~~add x12, x10, x11~~		
~~subi x13, x12, 16~~		

		Cycle1	Cycle2	Cycle3	Cycle4	Cycle5	Cycle6	Cycle7
1	or	IF	ID	EX	MEM	WB		
2	ld		IF	ID	EX	MEM	WB	
3	ld			IF	ID	EX	MEM	WB
4	add				~~IF~~	~~ID~~ IF	~~EX~~ ID	~~MEM~~ EX
5	subi				[illegible] ↓ stall	~~EX~~ ~~IF~~ ~~EX~~	~~ID~~ IF	EX ID

In cycle 4, Forward A = 10

In cycle 6, Forward B = 01

~~In cycle , Forward~~ -

Problem 8:

Solution:

	Instructions per program	CPI	Circuit Complexity
A	Decrease. Because this 3-operand ALU could combine 2 "add" into 1 instruction	Increase. May cause more hazards and increase the # of stalls	Increase. Add 1 more ALU
B	No effect. Do not affect the # of instructions	Increase. Because there are more stalls happens in this only ALU.	Decrease. Only need 1 ALU
C	No effect. Will not influence the # of instructions	No effect	Increase. Need extra hardware to support 64 register