Department of Computer Science
National Tsing Hua University
CS4100 Computer Architecture
Midterm, Nov. 21, 2016
2:20-4:20 PM

1. (10%) Explain the following terms:
   (a) Power Wall

由於能量的限制，造成效能提升的瓶頸 (沒提到校能扣 1.5 分)

   (b) Moore's Law

每 18 個月單位面積上電晶體數量 可以增加一倍

   (c) RISC vs. CISC (need to give processor examples)

RISC: Uses a small, highly optimized set of instructions, rather than a more versatile set of instructions often found in other types of architectures. E.g. ARM.
CISC: Single instructions can execute several low-level operations (such as a load from memory, an arithmetic operation, and a memory store) or are capable of multi-step operations or addressing modes within single instructions. E.g. Intel X86.

   (d) Amdahl's Law

The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. (textbook page 51)

2. (10%) Let the multiply instruction take 12 cycles and all other instructions 4 cycles. Now, given an embedded program with 15% multiply instructions. Suppose there is a new design where the multiply instruction can be reduced to 8 cycles but the cycle time increased by 20%. Let the embedded program be run on the old and new designs.
   (5%) What are the CPIs by the old and new designs?
   (5%) In terms of performance, which design is better (need to justify your answer)?

(a)

old design CPI = $12 \times 15\% + 4 \times 85\% = 1.8 + 3.4 = 5.2$

new design CPI = $8 \times 15\% + 4 \times 85\% = 1.2 + 3.4 = 4.6$

3. (10%) There are four instruction-set-architecture design principles: 1. Simplicity favors regularity, 2. Smaller is faster, 3. Make the common case fast, 4. Good design demands good comprises. Please give one MIPS design example for each of the principle

.

3.

(1). 32 bits per instruction ⇒ regularity

(2). 32 registers with each 32 bits ⇒ faster than memory

(3). immediate format make the constant calculate faster

(4). I: | opcode | rs | rt | imm. |

前面 format 盡量一樣

4. (10%) Please translate the following C code to MIPS assembly.

While (*A[j]* < *limit*)
  j++;

4

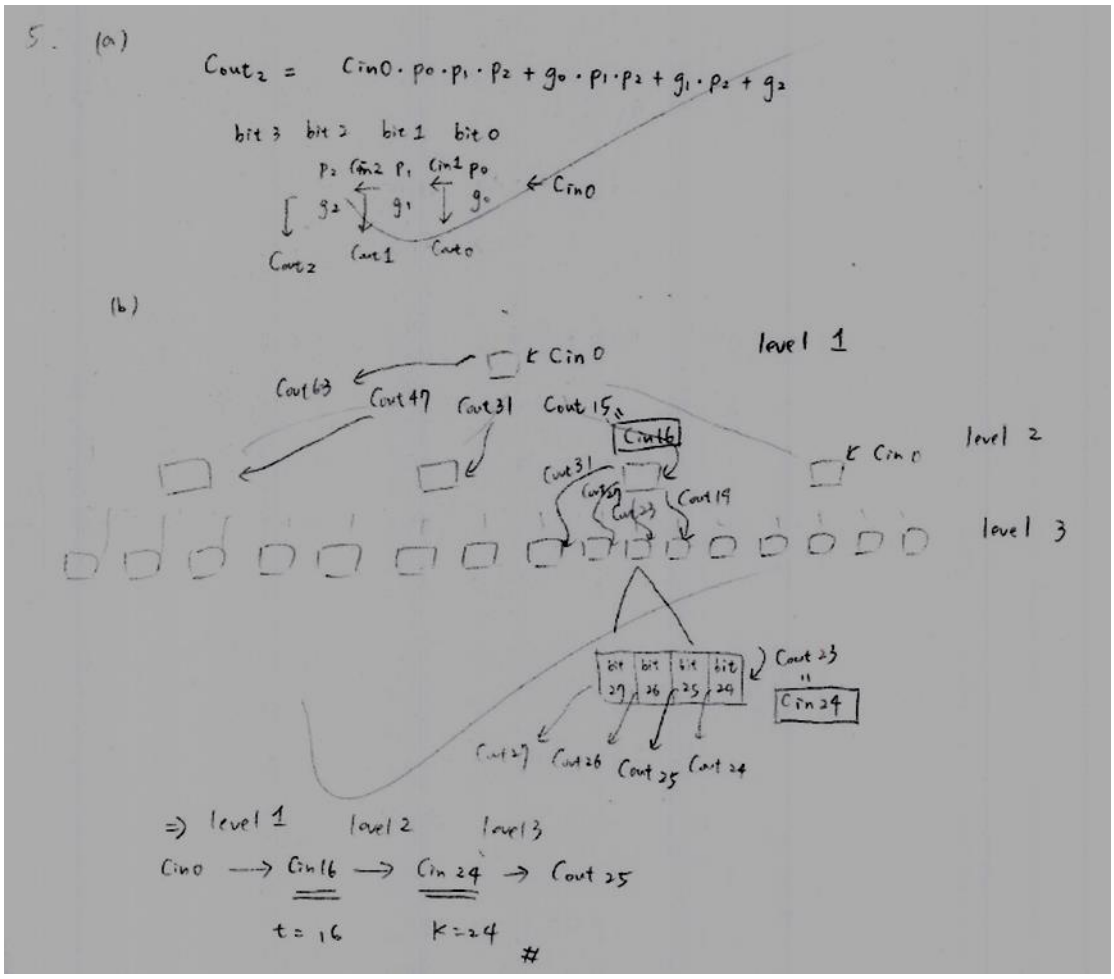
```
loop:
    mul $t0, $s1, 4
    lw $t1, $t0($s3)
    slt $t2, $t1, $s2
    beq $t2, $zero, Exit
    addi $s1, $s1, 1
    j loop

Exit:
```

5. (10%) Adder design

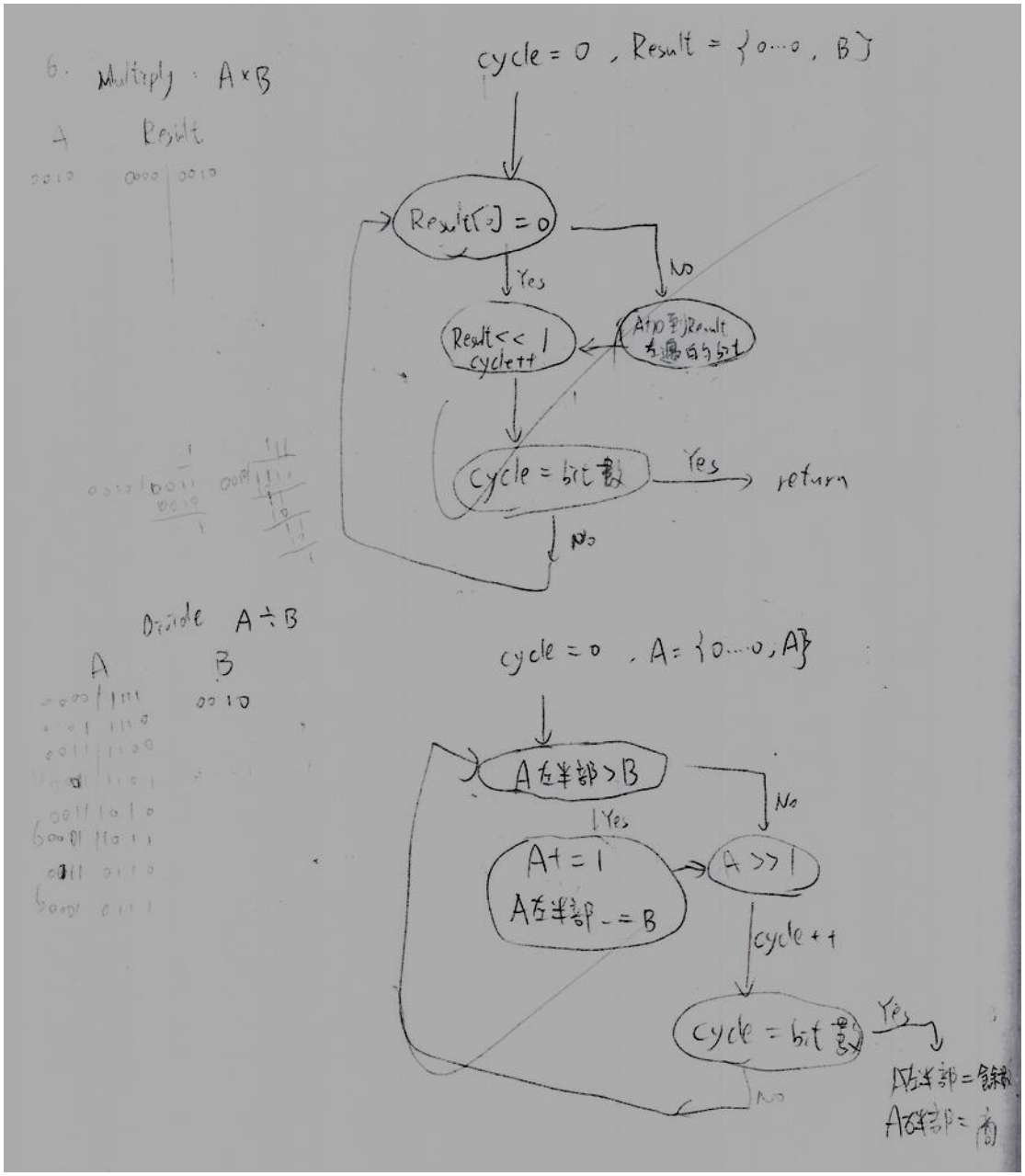
(a) (5%) Let us have a bock of 4-bit look-ahead adder. The least significant bit is a0/b0 and its carry-in is denoted as Cin0 and carry-out as $Cout_0$. That is, $Cout_i = Cin_{i+1}$. Let pi and gi be the propagate and generate terms of each bit *i* . Give the sum of product form to compute $Cout_2$ using pi, gi and Cin_0

(b) (5%) Design a 64-bit multiple level carry look-ahead adder with four-bit a block. We know that Pi and Gi to each block are produced from the bottom to the top of the tree while carry bits are produced from the top to the bottom. Two Cin_t and Cin_k must be produced before $Cout_{25}$ can be produced. What is *t* and *k*?



6. (10%) Multiply/Divide.

Give the block diagram of a sequential multiplier/divisor and explain how this function unit performs multiply and divide.



7. (20%) Consider the following instruction formats, R, and I types, for MIPS.

a. (5%) For R-type instruction, why 5-bit is used for the field of *shamt*?

b. (10%) For I-type instruction, consider the following MIPS code segment. To the left of each instruction we show the memory address (in base 16) where each instruction is stored.

Memory Address	Label	Instruction
0x1000 0010	L1:	add $s2, $s3, $s4
…		
0x1000 010C		bne $s2, $s5, L1

Below is the binary format for the *bne* instruction of the above code segment. Complete the binary representation of the instruction by writing the binary values for **rs** and **immediate**. (Recall that $17 = $s1)

31 opcode 26	25 rs 21	20 rt 16	15 immediate 0
000101			

c. (5%) Consider the above instruction. If $s2 = 0x00020000 and opcode is changed to a **load** instruction, what is the memory address of the loaded data?

(a) ∵ 一個 register 有 32 bits

∴ 可以 shift 0 bit or 1 bit or 2 bits or 3 bits....

31 bits

⇒ 總共有 32 種 shift 的方式

∴ shamt 需要 5 bits 去表示. ※

(b) $s2 = $18 → binary form : $10010_{(2)}$

∴ rs = 10010.

$55 = $21 → binary form : $10101_{(2)}$

∴ rt = 10101.

∵ PC 所指向的 memory address 是 0x10000110

L1 的 memory address 是 0x10000010

∴ $110_{(16)} = 256 + 16_{(10)} = 272_{(10)}$

$010_{(16)} = 16_{(10)}$

$16 - 272 = -256_{(10)}$

$\frac{-256}{4} = -64$ → binary form : 1111111111000000

⇒ immediate : 1111111111000000 ※

(c)

In decimal form, rs = 18, rt = 21, immediate = -64

instruction : lw $21 -64($18)

$52 = $18 的 base address = $0x00020000_{(16)}$

15 16

∴ 0x00020000 − 64 bytes = 0x0001FFC0.

⇒ the memory address of the loaded data

is 0x0001FFC0. ※

8. (10%) Define a new floating point format where bit 7 is a sign-bit, bits 6-4 an exponent, bits 3-0 a significant, bias 4 is used for exponent and hidden 1 is taken. Exponent = 000 and 111 are defined as special numbers the same as IEEE 754. Convert the following floating point numbers to decimal.

(1) 11101100

(2) 00000100

(1) sign bit = 1 ⇒ "negative"

exponent = $110_{(2)} \rightarrow 6_{(10)}$

$6 - 4 = 2$

∴ binary form : $1.11_{(2)} \times 2^2$

decimal form : $-(1 + 2^{-1} + 2^{-2}) \times 2^2 = -7$ #

(2) sign bit = 0 ⇒ "positive"

exponent = 000

significand = 0100

∴ It's a denormalized number.

⇒ binary form : $0.01_{(2)} \times 2^{-3}$

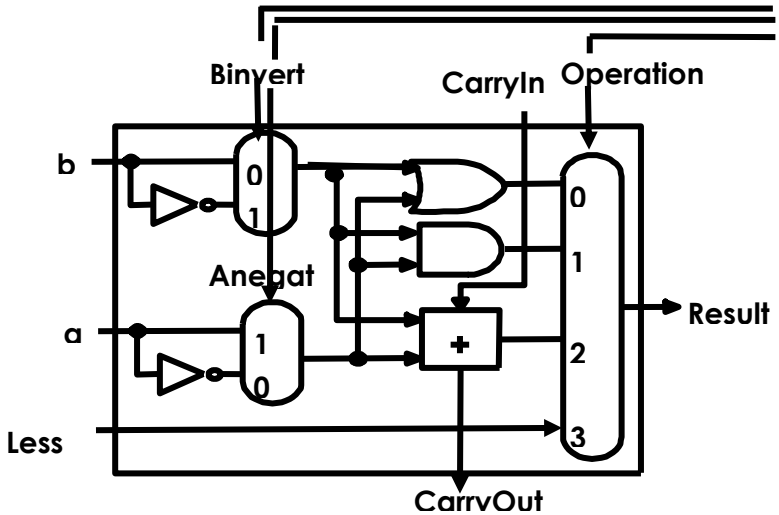
decimal form : $0.25 \times 0.125 = 0.03125$ #

9. (10%) The following figure shows an implementation of ALU. Please give the function specifications of the following operations.

Operations 4 bits (Anegat, Binvert, Operation)

a Sub *b*

a Nor *b*



a − b = a + (b' + 1)

∴ Anegat 選 1 , Binvert 選 1 , Operation = 10

ALU control of "a Sub b" = 1110.

a nor b = (not a) and (not b)

∴ Anegat 選 0 , Binvert 選 1 , Operation = 01

ALU control of "a Nor b" = 0101.

#