國立交通大學 資訊工程系 計算機組織 第一次考試 (2022/4/7 1:20PM~3:20PM)

請注意：1. 不可看書及任何參考資料(Close book)。 2. 請詳述設計或計算過程(detail your design or computation process)。 3. 請依題號作答，否則高題號答案出現後，之後的低題號答案將不予計分 (answer each problem in problem sequence. The answers of small ID problems that appear behind the answers of large ID problems will be ignored)。

1. Domain background (8 pts)

For a semiconductor manufacturing foundry, the die cost is determined by the following formula: *cost_per_die* = *cost_per_wafer*/(*dies_per_wafer*×*yield*), *dies_per_wafer* ≈*wafer_area*/*die_area*, *yield* = 1/(1+(*defects_per_area*×*die_area*/2))^2. For the business model of focusing on the manufacturing without designing chip layout for the customers, please analyze why TSMC has benefits of die cost as compared to the other foundries.

Since the die area is not designed by TSMC, die area is a constant for TSMC. The factor TSMC can control is the defects per area and TSMC has very small defects per area as compared to the other foundries. So TSMC has very high yield. Dies per wafer is also a constant for TSMC (TSMC cannot change this value). The cost per wafer also can be regarded a constant. So finally the cost per die for TSMC is determined by the yield only, and then TSMC can outperforms the other foundries.

(評分標準:提及良率高導致成本下降即可全拿，但若講到因為製程可以做到很小蕙扣 4 分)

2. Performance (23 pts)

The table below shows the alternatively compiled code sequences using instructions in classes *A*, *B* and *C*. number of executed instructions and the average CPI for running applications 1 and 2 on two machines. The clock rates of machines *A* and *B* are 4 GHz and 2 GHz. (*must show all computations to get full credits*)

class	A	B	C
CPI for class	1	2	4
IC in sequence 1	2	3	3
IC in sequence 2	4	1	1

(a) (5 pts) Which code sequence is faster and by how much?

(b) (8 pts) In CMOS IC technology, power consumption is computed by the following formula: *P*=*capacitive_load*×*frequency*×*voltage*^{2}. Now we plan to design a new CPU that can run the code sequence 1 with the same time as that of the code sequence 2 on the old CPU. The new CPU has the same ISA as the old one and is asked to run with the same power consumption as the old one. If

we have 12.5% capacitive load increase in the new design, what many voltage reduction and how many clock frequency increase the new CPU should have?

(c) (5 pts) Consider the performance metric MIPS. List the formula to compute the MIPS. Which kind of CPU has the superiority if we use MIPS to evaluate the performance of a CPU. You have to use the MIPS formula to explain your reason to get full credit.

(d) (5 pts) A CPU has two function units *A* and *B*. A program requires 10 secs for *A* and 20 secs for *B*. Now we want to accelerate the program by refining function unit *A* only. Can we reduce the runtime of the program to the half of original time on the new CPU? Why?

a. 1: clock cycle = 2*1 + 3*2 + 3*4 = 2 + 6 + 12 = 20.
2: clock cycle = 4*1 + 1*2 + 1*4 = 10. CS 2 is faster by 20/10 = 2.

b. New cpu frequence: 20*cycle_time_new = 10*cycle_time_old
Cycle_time_new = 0.5 cycle_time_old, frequence_new = 2.0 frequency_old
$1.125CL*2clock*newVol^2 = CL*clock*oldVol^2$, $newVol = (1/2.25)^{0.5} oldVol$
newVol = (1/1.5)oldVol=0.67oldVol, 33% voltage reduction

c. MIPS = instruction count/(execution time*10^6) = clock rate/(CPI*10^6). RISC CPU has the superiority because RISC cpu has smaller CPI and generally can run in higher clock rate but may have larger instruction count. In this case larger instruction count does not affect the MIPS and can have larger MIPS due to smaller CPI and higher clock rate.

d. No. Original runtime is 30 secs. The half is 15 secs. We do not refine B, so the time for B is still 20 secs. So we cannot reduce the runtime by half if we refine function unit B only.

3. Design Principles (12 pts)

List three design principles for the instruction set of MIPS CPU. Also use the instruction set as an example to explain the meaning of each principle.

A. Simplicity favors regularity. MIPS does not support any form of addition, and it only supports the simplest addition, two input operands and an output. All addition must be decomposed into several two input-operand additions. This regaularity makes the implementation simple and simplicity enables higher performance at lower cost.

B. Smaller is faster. Register files only support 32 registers but not many register, which makes the design of register file simple and then register access becomes faster than a register files with many registers.

C. Make the common case fast. MIPS support the immediate constant in an add/sub instruction to diminish the memory access in some instructions that are frequently used. Although this scheme increases the complexity of instruction format a little bit, this scheme makes the common cases faster.

D. Good design demands good compromises. I-format instructions, such as beq, make the instruction set complex, but offer diverse functions as well, a good compromise.

```
Input: two unsigned numbers
in $t1 and $t2
addu $v0, $t1, $t2
nor  $t3, $t1, $0
sltu $t4, $t3, $t2
beq  $t4, $0, L1
sub  $v1, $t2, $t3
j    L2
L1: add $v1, $0, $0
L2: jr $ra
```

Code seq 1

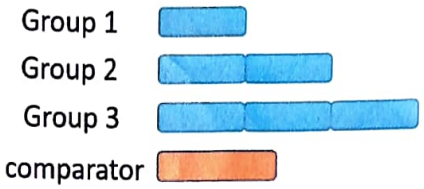


Fig. 1

4. Instructions and decoding/encoding (28 pts)

Assume the initial registers are all 0, and consider the code sequence 1 below.

(a) (6 pts) What does the code sequence 1 do?

(b) (8 pts) What is the binary number of instruction "beq $t4, $0, L1", where the number of $t4 is 12 (4 pts)? What is the associated MIPS instruction of the binary instruction, 10000010000100010000000000000001 (4 pts)?

(c) (6 pts) MIPS CPU only support beq and bne instructions, and does not have the other type of comparison instructions, like bgt, bge, blt, and ble. Suppose that MIPS CPU already have three groups of instructions, as shown in Fig. 1, where group *n* implies that the instructions in group *n* require *n* clock cycles to complete their execution. Fig. 1 also shows the required number of clock cycles for a comparator. Explain the main reason why MIPS CPU does not support all types of comparison based on the scenario in Fig. 1.

(d) (8 pts) A C code and its related MIPS assembly codes are shown as follows. Please explain why we need to store/restore the values of $ra and $a0 in the procedure call (4 pts). As we invoke the procedure by fact(2), list the execution sequence, like the sequence (1, 2, 3, 4, 5, 6, 7, 8, 9) for fact(0) (4 pts).

MIPS assembly				C codes
1	fact:	9	jr $ra	int fact(int n)
2	addi $sp, $sp, -8	10	L1: addi $a0, $a0, -1	{
3	sw $ra, 4($sp)	11	jal fact	if (n < 1) return (1);
4	sw $a0, 0($sp)	12	lw $a0, 0($sp)	else return (n*fact(n-1));
5	slti $t0, $a0, 1	13	lw $ra, 4($sp)	}
6	beq $t0, $zero, L1	14	addi $sp, $sp, 8	
7	addi $v0, $zero, 1	15	mul $v0, $a0, $v0	
8	addi $sp, $sp, 8	16	jr $ra	

a. This code sequence performs an unsigned addition and return the addition result in $v0. It also checks if an overflow occurs. If an overflow occurs, the excessive amount is returned in $v1; otherwise, return 0 in $v1. ×2 ×2 1pt

b. 000100 01100 00000 0000000000000010 (一個欄位一分)

lb $t3, 1($t4)

c. If all types of comparisons are supported, then MIPS requires a comparator that requires more than 1 clock cycle and less than 2 clock cycles for its execution. Then performing the instructions using the comparator requires two clock cycles for execution and there will be some idle time. On the contrary, if we increase the clock cycle time to fit the required time of the comparator, then the instructions in groups 1 and 2 have some idle time to fit the new clock cycle time. All ways will have idle time to lower the CPU performance.

d. $ra: this register stores the return address of this call. As the procedure acts as a caller to call the other procedure, $ra will be replaced as a new value. So we need to save current return address and restore it as the new call ends.
$a0: This register stores the input parameter to the callee. As we want to call another procedure in this procedure, we need to save the input parameter first and then call another procedure; otherwise, the input parameter to this procedure will get lost.

(1(n=2), 2, 3, 4, 5, 6, 10, 11, 1(n=1), 2, 3, 4, 5, 6, 10, 11, 1(n=0), 2, 3, 4, 5, 6, 7, 8, 9, 12 (n=1), 13, 14, 15, 16, 12 (n=2), 13, 14, 15, 16) (有寫(n=x)的地方到下一個(n=x)的區間可視為一個區域，照理說整個區域應該是全對或全錯(因為中間都是循序執行不會跳躍執行)，錯的就扣一分，扣到零分為止)

5. Carry Look-Ahead (CLA) Adder (14 pts) Consider a 16-bit CLA composed of several 4-bit CLAs. The generate and propagate functions are $gi = a_i \times b_i$ and $p_i = a_i + b_i$.

(a) (4 pts) What are the functions of group generate $G_{4i\text{-}4i+3}$ and group propagate $P_{4i\text{-}4i+3}$ for i=0 to 3?

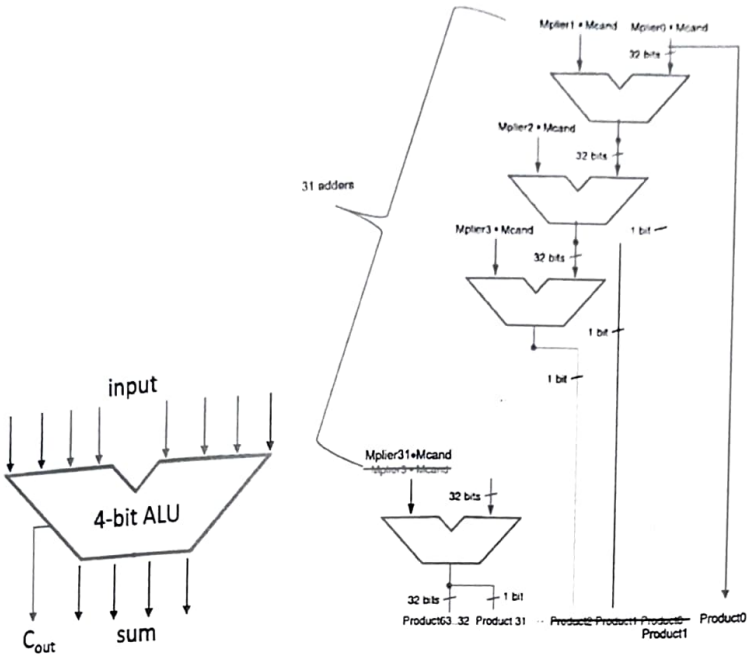
(b) (5 pts) Design the multi-level 16-bit CLA.

(c) (5 pts) Assume that the delays of XOR gate, AND-OR compound gate and single AND gate are all 2 delay units. What is the delay of the n-level 4^n-bit CLA? (you must list the complete formula to get the credit. No credit will be given if no detailed formula and related explanation is given)

a. group propagate $P_{4i\text{-}4i+3} = \Pi p_j$ for j=4i to 4i+3, $G_{4i\text{-}4i+3} = \Sigma((\Pi p_k$ for $k = j+1$ to $4i+3) \times g_j)$ for $j = 4i$ to $4i+3$

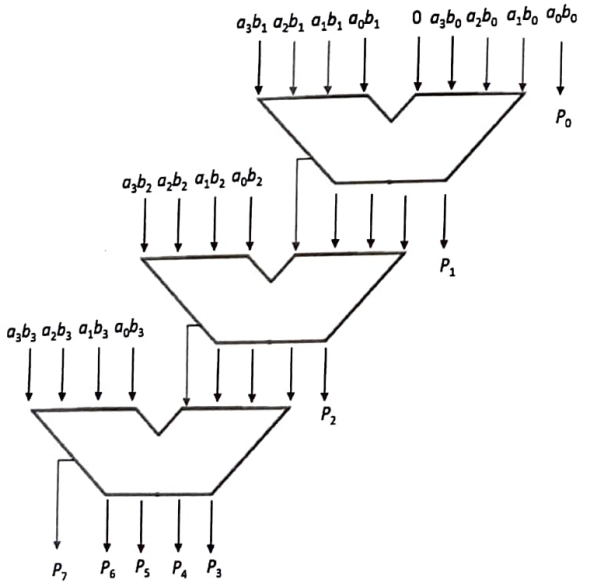
b. page 23

c. 2 (G and P) + 2n (group G and P during downward pass) + 2(n-1) (group carry in during upward pass) + 2 (msb sum) = 4n+2



6. Fast Multiplier (8 pts)

The right figure above shows a fast multiplier that does not require storing intermediate addition result to a register. But this figure is only to highlight the concept. For detailed design, we still need to make a correct bit alignment for two input operands of each ALU. The left figure above shows a 4-bit ALU with individual wire for each bit input/output. Use this module to realize a fast multiplier to complete $a_3a_2a_1a_0 \times b_3b_2b_1b_0$. You need to specify the signal name to each bit and you also need to complete the wiring between two connected ALU at bit-level.



7. Floating-point number and arithmetic (12 pts)

IEEE standard 754 defines the floating-point number standard, where the single-precision floating point number is 32-bit long with 8-bit exponent, 23-bit fraction, and an exponent bias of 127.

(a) (6 pts) What are the least positive single-precision normalized and de-normalized numbers?

(b) (6 pts) Consider the schemes of guard bit, round bit, and sticky bit. Now we use "digit" rather than "bit" in the following problem. Each fraction is rounded off to the second decimal place. What are the accuracies of the addition $2.35\times10^{1} + 7.51\times10^{-1}$ with and without guard, round, and sticky digits in terms of ulp?

a. $0\ 00000001\ 0\ldots0 = 1.0 * 2^{-126}$. $0\ 00000000\ 0\ldots01 = 1.0* 2^{-23}* 2^{-126}=1.0*2^{-149}$

b. With the scheme: 0.49 ulp. Without the scheme: 0.51 ulp.

Op (31:26)								
28-26 31-29	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	R-format	Bltz/gez	Jump	Jal	Beq	Bne	Blez	Bgtz
1(001)	Addi	Addiu	Slti	Sltiu	Andi	Ori	Xori	Lui
2(010)	TLB	FIPt						
3(011)								
4(100)	Lb	Lh	Lwl	Lw	Lbu	Lhu	lwr	
5(101)	Sb	Sh	Swl	Sw			Swr	
6(110)	Lwc0	Lwc1						
7(111)	Swc0	swc1						

Op (31:26)=010000 (TLB), rs(25:21)								
23-21 25-24	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(00)	mfc0		cfc0		mtc0		ctc0	
1(01)								
2(10)								
3(11)								

Op(31:26)=000000 (R-format), funct(5:0)								
2-0 5-3	0(000)	1(001)	2(010)	3(011)	4(100)	5(101)	6(110)	7(111)
0(000)	Sll		Srl	Sra	Sllv		Srlv	Srav
1(001)	Jump reg.	Jalr			Syscall	Break		
2(010)	Mfhi	Mthi	Mflo	Mtlo				
3(011)	Mult	Multu	Div	Divu				
4(100)	Add	Addu	Subtract	Aubu	And	Or	Xor	Nor
5(101)			Set l.t					
6(110)								
7(111)								