國立交通大學 資訊工程系 計算機組織 第一次考試 (2021/4/15 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 (5 pts)**
   Workstations were the main computers for high-performance applications in many areas, but now workstations had been weeded out. Which kind of computers replace the role of workstations (2 pts)? How can this happen (3 pts)?

2. **CPI and workload (20 pts)**
   The table below shows the 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*)

	Machine A		Machine B	
	Instructions	Avg CPI	instructions	Avg CPI
Application 1	2.0E+9	2	1.0E+9	4
Application 2	4.0E+9	2.5	1.5E+9	3

   (a) (5 pts) If application 2 must run two times as often as application 1, which machine is faster?

   (b) (5 pts) Only consider the performance of running application 2, which machine is better in terms of MIPS (3 pts)? Use this result to explain the reason why MIPS is (or is not) a fair performance comparison metric (2 pts).

   (c) (5 pts) We refine the floating-point (FP) unit of machine *A* and keep the other units unchanged. If the application 2 has half of instructions associated with FP operations, and the average CPI of all FP instructions in application 2 is diminished from 4 to 3 through the refinement engineering. Which machine has better performance in application 2?

指令(2个) (d) (5 pts) If we try to better machine *B*'s performance by raising the clock rate, use one example in figure to illustrate why the purpose may not be satisfied in this way.

3. **Instructions and decoding/encoding (26 pts)**
   Assume the initial registers are all 0, and consider the code sequence 1 below.

   (a) (5 pts) What does this code sequence do?

   (b) (8 pts) What is the binary number of instruction "beq $s0, $0, L1", where the number of $s0 is 16 (4 pts)? What is the associated MIPS instruction of the binary instruction, 10000010000100010000000000000001 (4 pts)?

   (c) (8 pts) Please point out the location of the code sequence 2 below to violate the procedure calling convention of MIPS (4 pts) and correct the program to fix the convention violation (4 pts).

(d) (5 pts) Explain what the code sequence 3 does.

```
lui  $t0, 0x2409
ori  $t0, $t0, 0xa1d0
lw   $t1, 0($t0)
lw   $t2, 4($t0)
sub  $s1, $t2, $t1   //s1=t2-t1
slti $s0, $s1, 0     // if s1<0, s0=1
beq  $s0, $0, L1     //if s0=0, L1
sub  $s1, $t1, $t2   // s1=t1-t2
L1: sw  $s1, 8($t0)
```

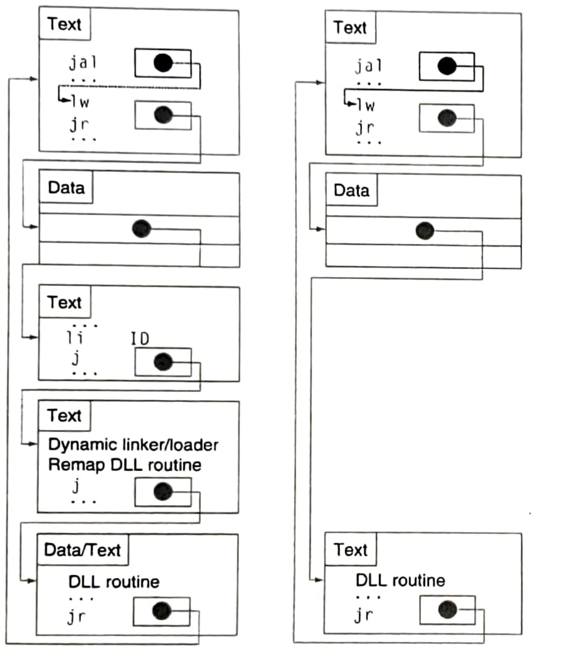
Code seq 1

```
Leaf1:  addi $sp, $sp, -4
  sw    $s0, 0($sp)     // s0 → sp[0].
  add   $t0, $a0, $a1   t0 = a0+a1
  add   $t1, $a2, $a3   t1 = a2+a3
  sub   $s0, $t0, $t1   S0 = t0-t1
  add   $v0, $s0, $zero V0 = S0 + zero
  jal   Leaf2
  lw    $s0, 0($sp)
  addi  $sp, $sp, 4
✓ jr    $ra
```

Code Seq 2

```
try:
add $t0,$zero,$s0   t0 = S0.
ll $t1,0($s2)
sc $t0,0($s2)
beq $t0,$zero,try
add $s0,$zero,$t1
```

Code seq 3



a. First call to DLL routine b. Subsequent calls to DLL routine

4. **Lazy linkage (10 pts)**

Explain how the lazy linkage works.

5. **32-bit ALU design (6 pts)**

Consider the 32-bit ALU design with *slt* instruction, please describe how to realize the *slt* instruction in the ALU. You do not need to draw the block design, just describe your design concept in texts, including the setting of control signals to perform specific operations.

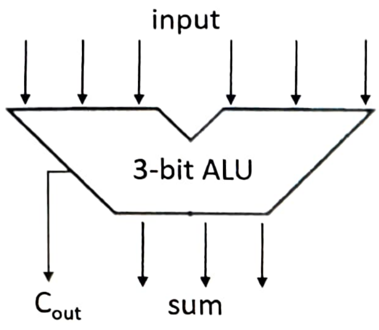
6. **Carry Look-Ahead (CLA) Adder (15 pts)** Consider a 16-bit CLA composed of several 4-bit CLAs. The generate and propagate functions are $gi = a_i \times b_i$ and $p_i = a_i + b_i$. $C_i = g_i + P_i C_{i-1}$

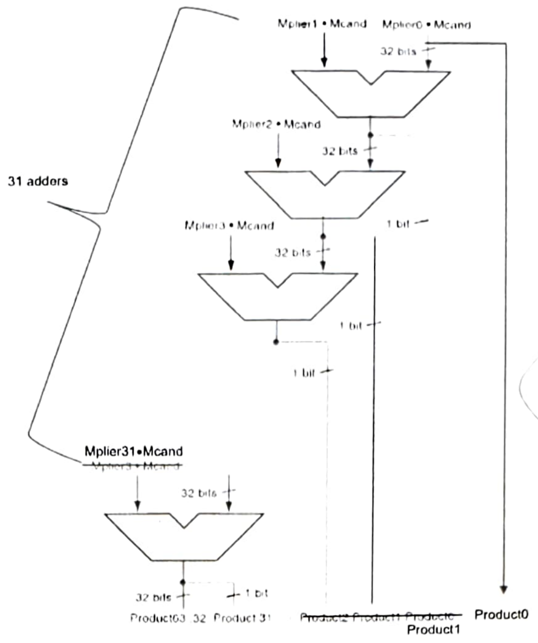
(a) (4 pts) What are the functions of group generate $G_{4i\text{-}4i+3}$ and group propagate $P_{4i\text{-}4i+3}$ for i=0 to 3?

(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 multi-level 16-bit CLA?

if input 不是另一个 output 三個求

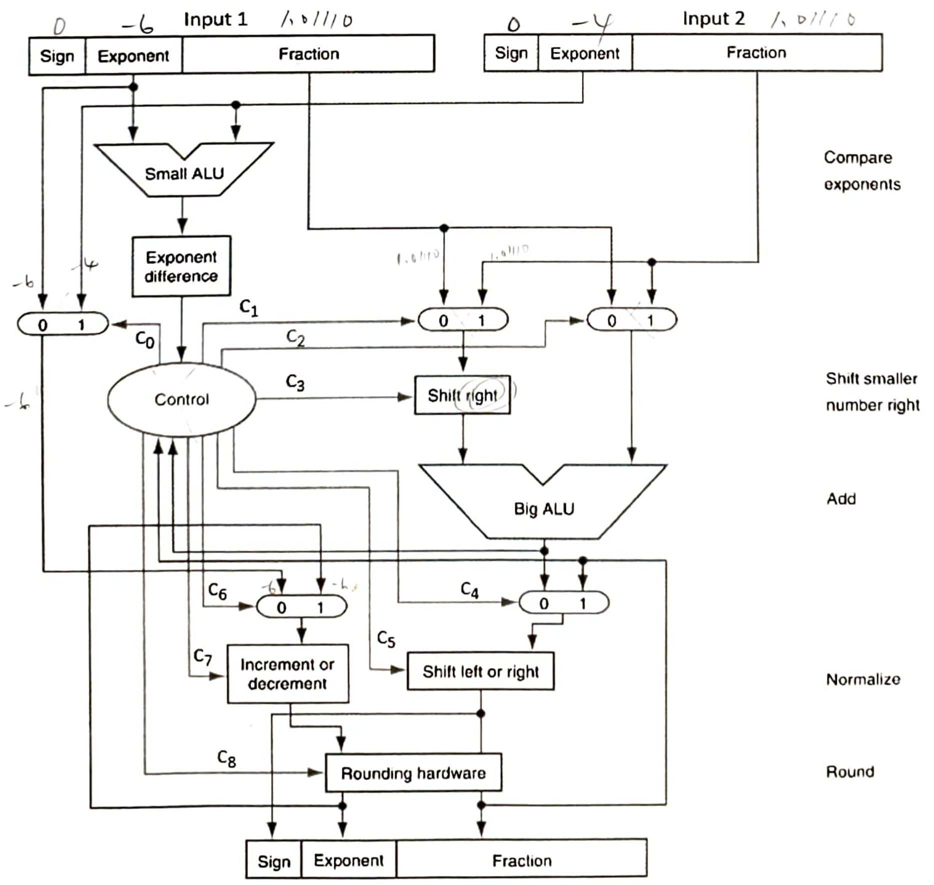




細緻化

7. **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 3-bit ALU with individual wire for each bit input/output. Use this module to realize a fast multiplier to complete $a_2a_1a_0 \times b_2b_1b_0$. 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.



8. Floating-point number and arithmetic (11 pts)

(a) (5 pts) The above figure depicts a floating-point adder design with guard and round bits to improve accuracy. You cannot see the guard and round bits in the design, just imagine there are two hidden bits behind the design and it indeed works. The adder uses the policy of rounding. Consider the addition of two binary floating-point numbers, 1.01110×2^{-6} (input 1) and 1.01110×2^{-4} (input 2). Here we do not consider the exponent bias. List the value of each control signal. For multiplexers, list the value of each control signal to select correct input. For the other hardware, describe the operation that will be performed by each hardware.

(b) (3 pts) What is the final addition result of this adder?

(c) (3 pts) What is the accuracy of this addition result as compared to the correct value in terms of 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	FlPt						
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)								