國立交通大學 資訊工程系 計算機組織 第一次考試 (2014/4/10 10:10AM~12:20PM)

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. **Performance and CPU Clocking Frequency (9%)**
   - **a、** (3%) Consider a CPU design, does a CPU with higher clocking frequency necessarily run faster than using a lower clocking frequency (the same CPU)?
   - **b、** (6%) If your answer to (a) is YES, please **explain your answer** for YES, and **use a counter-example** to explain your answer for NO.
2. **Performance and CPI (18%)**
   - **a、** (3%) Two programs *A* and *B* realize two different algorithms for the same problem and are compiled with two different compilers to run on the same computer. Program *A* has smaller CPI than program *B*. Does program *A* necessarily run faster than program *B*?
   - **b、** (6%) Please explain your answer to (a) **in terms of the number of clock cycles** required to run a program.
   - **c、** (9%) Consider the following table that lists CPIs for three instructions and two code sequences compiled by two compilers. For instance, the first code sequence contains 2 A-type instructions, 8 B-type instructions, and 3 C-type instructions. Which code sequence has better performance (3%)? If we want to improve the hardware design realizing B-type instructions, what should the new CPI of B-type instructions be after our hardware refinement to make two code sequences have the same performance (3%)? Can we reach the same goal by refining the hardware part that realizes C-type instructions rather than B-type instructions (3%, list complete derivation process to get full credit)?

Class	A	B	C
CPI for class	3	4	2
I.C. in sequence 1	2	8	3
I.C. in sequence 2	6	3	2

3. **MIPS (9%)**
   - **a、** (3%) Millions of Instructions Per Second (MIPS) is a metric to evaluate computer performance. Please list its formula.
   - **b、** (6%) Use MIPS's formula to illustrate why MIPS is not a precise metric to evaluate computer performance.
4. **Design principles (12%)**

   Please list four design principles (including one example for illustrating each principal) for MIPS CPU design.
5. **Procedure call (12%)**

The following program with 11 instructions is not correct. Modify the program to make it right

instruction	Instr. address	Instruction format					
(1) strcpy: add $s0, $zero, $zero	80000						
(2) L1: add $t1, $s0, $a1	80004						
(3) lbu $t2, 0($t1)	80008						
(4) add $t3, $s0, $a0	80012						
(5) sb $t2, 0($t3)	80016						
(6) beq $t2, $zero, L2	80020				ADR1		
(7) addi $s0, $s0, 1	80024				1		
(8) j L1	80028		ADR2				
(9) L2: lw $s0, 0($sp)	80032						
(10) jal PRT_MSG	80036		ADR3				
(11) jr $ra	80040	0	31	0			8
As procedure strcpy is invoked, initial registers are set as $a0 = 20000, $a1 = 24000, $sp = 60000							

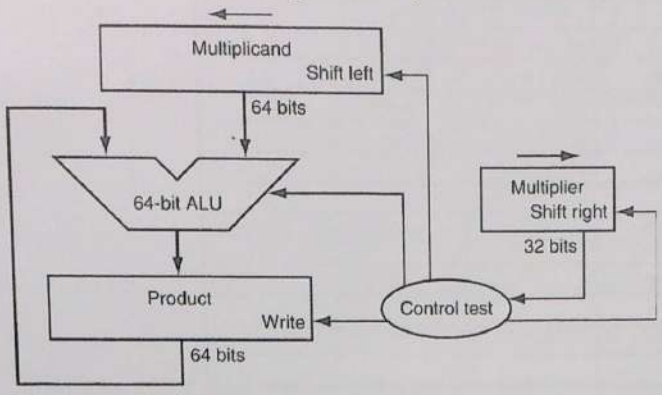
6. **Instruction Format and Addressing Mode (15%)** (Answer this problem based on the program in the above table instead of your answer for Problem 5)
   - **a、** (6%) The above table shows a MIPS program. What are the values of ADR1 and ADR2? Notably, the final instruction address is obtained by concatenating partial value of PC register with that of instruction address (column 2)
   - **b、** (9%) Please complete the machine codes of instructions (2), (3), and (9). The instruction code table is listed in the last page (P. 4). The register numbers for $t1, $s0, $a1, and $sp are 9, 16, 5 and 29 respectively.
7. **Synchronization and Linkage (12%)**
   - **a、** (6%) Please describe the operations of two instructions, Load linked: ll rt, offset(rs) and Store conditional: sc rt, offset(rs).
   - **b、** (6%) Explain the following code sequence (use "//" to separate adjacent instructions): try: add $t0,$zero,$s4 // ll $t1,0($s1) // sc $t0,0($s1) // beq $t0,$zero,try // add $s4,$zero,$t1
8. **Addition overflow (8%)**

   The following two code sequences are to detect the overflows in signed and unsigned additions. Two operands are stored in $t1 and $t2. Please complete two code sequences.

Signed addition	Unsigned addition
addu $t0, $t1, $t2 Incomplete Inst1 slt $t3, $t3, $zero bne $t3, $zero, No_overflow Incomplete Inst2 slt $t3, $t3, $zero bne $t3, $zero, Overflow	addu $t0, $t1, $t2 Incomplete Inst3 Incomplete Inst4 bne $t3, $zero, Overflow

9. **Multiplier hardware (6%)**

The following figure is a multiplier design. Please propose a refined version to lower the hardware cost. Describe the multiplication procedure based on the proposed hardware.



10. **Bonus (8%)** For each pseudoinstruction in the following table, produce a **minimal sequence** of actual MIPS instructions to accomplish the same thing. In the following table, big refers to a specific number that requires 32 bits to represent.

Note: (a). you can use upper16(big) to represent the upper 16 bits of big; and lower16(big) to represent the lower 16 bits of big. (b). **Please use the register specifically reserved for assembler**, if necessary. (c). For MIPS assembly instructions, please refer to the table on the last page.

	Pseudoinstruction	What it accomplishes	(Hint: Minimal sequence)
1 (2%)	bge $t5, $t3, L	if($t5 >= $t3) goto L	2 instructions
2 (2%)	bgt $t5, $t3, L	if($t5 > $t3) goto L	2 instructions
3 (2%)	Li $t5, big	$t5 = big	2 instructions
4 (2%)	lw $t5, big($t2)	$t5 = Memory[$t2+big]	4 instructions

Op (31:26)								
26-26 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.					