# 國立交通大學 96 學年度碩士班考試入學試題

科目：計算機組織(1006)　　考試日期：96 年 3 月 17 日 第 4 節
系所班別：資訊學院聯招　　組別：資訊聯招　　第 1 頁,共 3 頁

＊＊作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！

注意：1. 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
2. 如有運算需要，請詳細列出完整過程，否則不予計分
3. 題號請標示清楚，各題間請以橫線或空行分隔，答案請圖示或劃註底線

1. (Choice) (18 %) (Scoring policy is as follows. Final score = (# of your correct choices – # of your wrong choices) × (18 ÷ total number of correct choices of the solutions). For example, the solutions of this problem contain totally 10 correct choices and your correct choices and wrong choices are 9 and 5, respectively. Then you'll get (9 - 5) × (18 ÷ 10) = 7.2. Minimum score is 0 pt.)

(a) Which is (are) correct?

i. Suppose there was a 16-bit IEEE 754-like floating-point format with 5 exponent bits. (±1.0000 0000 00 × $2^{-15}$ to ±1.1111 1111 11 × $2^{14}$, ±0, ±∞, NaN) is the likely range of numbers it could represent.

ii. For 32-bit IEEE 754 floating-point standard, the smallest positive normalized number is: 1.0000 0000 0000 0000 0000 000 × $2^{-125}$.

iii. For 32-bit IEEE 754 floating-point standard, the smallest denormalized number is: 0.0000 0000 0000 0000 0000 001 × $2^{-126}$.

(b) Some programming languages allow two's complement integer arithmetic on variables declared byte and half word, i.e., 16 bits. What MIPS instructions would be used?

i. Load with *lbu*, *lhu*; arithmetic with *add*, *sub*, *mult*, *div*; then storing using *sb*, *sh*.

ii. Load with *lb*, *lh*; arithmetic with *add*, *sub*, *mult*, *div*; then storing using *sb*, *sh*.

iii. Loads with *lb*, *lh*; arithmetic with *add*, *sub*, *mult*, *div*; using *and* to mask result to 8 or 16 bits after operation; then store using *sb*, *sh*.

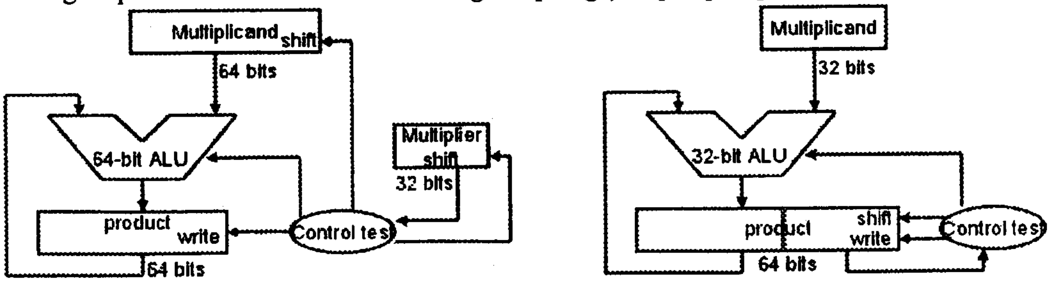
(c) Carry look-ahead adder can diminish the carry delay which dominates the delay of ripple carry adder. *Generate* ($g_i$) and *propagate* ($p_i$) functions are two main operations of carry look-ahead adder. Assume *a* and *b* are two operands and $c_{i+1}$ is the carry out of level *i* and carry in of level *i+1*, which is (are) correct?

i. $g_i = a_i \cdot b_i$

ii. $p_i = (a_i + b_i) \cdot c_i$

iii. If $g_i$ equals to 1, we can say the carry out of level *i* is 1.

iv. Carry look-ahead adder can be extended to multi-level style. The first *group generate* of a 3-bit group can then be defined as $G_0 = g_2 + (p_2 \cdot g_1) + (p_2 \cdot p_1 \cdot g_0)$



Structure a　　Structure b

(d) The above figure shows two multiplication structures. Which is correct?

i. The shift operation in the multiplicand in structure a is shift-right.

ii. The shift operation in the multiplier in the structure a is shift-right.

iii. The multiplier is stored in the right part of the product register in structure b.

iv. In structure b, one control signal for shifting multiplicand register is missed.

(e) About the 32-bit MIPS instructions, which description is correct?

i. MIPS has 32 registers inside CPU because it is a 32-bit CPU.

ii. *add* instruction can not directly store the addition result to memory.

iii. Since memory structure is byte-addressing, the address offset in *beq* instruction is referred to as byte.

iv. In MIPS, "branch-if-less-than" is realized using *slt* and *beq/bne,* since its design principle is two faster instructions are more useful than one slow and complicated instruction.

2. (8%) (a) (4%) What is procedure frame? Also, stack pointer and frame pointer are used to maintain procedure frame. Why does procedure frame require two pointers?

   (b) (4%) Procedure has to spill registers to memory (save and then restore). Caller must take care of $ax series and $tx series and callee must take care of $ra and $sx series. Following codes require correction for spilling registers. Correct the errors and state your reasons.

```
fact:                                  L1:   addi   $a0, $a0, -1
          addi   $sp, $sp, -4                jal    fact
          sw     $ra, 0($sp)                 lw     $ra, 0($sp)
          slti   $t0, $a0, 1                 addi   $sp, $sp, 4
          beq    $t0, $zero, L1              mul    $v0, $a0, $v0
          addi   $v0, $zero, 1               jr     $ra
          addi   $sp, $sp, 4
          jr     $ra
```

3. (5%) Please explain the concept of non-restoring division algorithm.

4. (10%) We wish to compare the performance of two different computers: M1 and M2. Following measurements have been made on these computers: Program 1 executes for 2.0 seconds on M1, and 1.5 seconds on M2, whereas Program 2 executes for 5.0 seconds on M1, and 10.0 seconds on M2.

   (a) (4%) Which computer is faster for each program, and how many times as fast is it?

   The following additional measurements were then made: Program 1 executes $5\times10^9$ instructions on M1, and $6\times10^9$ instructions on M2.

   (b) (2%) Find instruction execution rate (instructions/second) for each computer when running Program 1.

   Suppose M1 costs $500 and M2 costs $800. A user requires that Program 1 must be executed 1600 times each hour. Any remaining time is used to run Program 2. If the computer has enough performance to execute Program 1 the required number of times per hour, then performance is measured by the throughput for Program 2.

   (c) (2%) Which computer is faster for this workload? Why?

   (d) (2%) Which computer is more cost-effective? Show your calculations.

5. (9%) Given the code sequence:

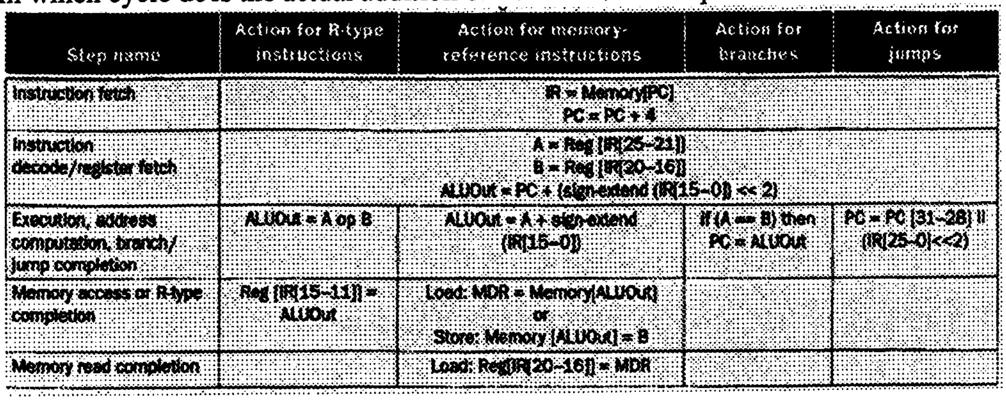
```
        lw   $t1, 8($t7)           ; assume mem($t7+8) contains (+72)10
        addi $t2, $zero, #10
        nor  $t3, $t1,   $t2
        beq  $t1, $t2,   Label
        add  $t4, $t2,   $t3
        sw   $t4, 108($t7)
Label:  ...
```

   According to the multi-cycle implementation scheme in the textbook (see figure below),

   (a) (3%) How many cycles will it take to execute this code?

   (b) (3%) What is going on during the 19th cycle of execution?

   (c) (3%) In which cycle does the actual addition of 108 and $t7 take place?



| Step name | Action for R-type instructions | Action for memory-reference instructions | Action for branches | Action for jumps |
|---|---|---|---|---|
| Instruction fetch | IR = Memory[PC]<br>PC = PC + 4 | | | |
| Instruction decode/register fetch | A = Reg [IR[25–21]]<br>B = Reg [IR[20–16]]<br>ALUOut = PC + (sign-extend (IR[15–0]) << 2) | | | |
| Execution, address computation, branch/jump completion | ALUOut = A op B | ALUOut = A + sign-extend (IR[15–0]) | if (A == B) then PC = ALUOut | PC = PC [31–28] II (IR[25–0]<<2) |
| Memory access or R-type completion | Reg [IR[15–11]] = ALUOut | Load: MDR = Memory[ALUOut]<br>or<br>Store: Memory [ALUOut] = B | | |
| Memory read completion | | Load: Reg[IR[20–16]] = MDR | | |

6. (6%) Instruction count, CPI, and clock rate are three key factors to measure performance. The performance of a program depends on the algorithm, the programming language, the compiler, the instruction set architecture, and the actual hardware used.
   (a) (3%) What performance factor(s) above may be affected by using different Instruction Set Architectures? Why?
   (b) (3%) MIPS (Million Instructions per Second) of running a benchmark program on machine A is higher than that of running the same benchmark on machine B. Which machine is faster? Why?

7. (10%) To implement these five MIPS instructions: [lw, sb, addi, xor, beq],
   (a) (3%) If simple single-cycle design is used, at least how many adders must be used? What each of these adders is used for?
   (b) (3%) Similarly, at least how many memories are there? What each of them is used for?
   (c) (2%) Repeat (a) for multi-cycle design.
   (d) (2%) Repeat (b) also for multi-cycle design.

8. (12%) Assume the three caches below, each consisting of 16 words. Given the series of address references as word addresses: 2, 3, 4, 16, 18, 16, 4, 2. Please label each reference as a hit or a miss for the three caches (a), (b), and (c) below. Assuming that LRU is used for cache replacement algorithm and all the caches are initially empty.
   (a) (4%) a direct-mapped cache with 16 one-word blocks;
   (b) (4%) a direct-mapped cache with 4 four-word blocks;
   (c) (4%) a four-way set associative cache with block size of one-word.

9. (8%) Continued from above question 8:
   (a) (2%) For each of above (a), (b), and (c) caches, how many misses are compulsory misses?
   (b) (2%) For each of above (a), (b), and (c) caches, how many misses are conflict misses?
   (c) (2%) What type of cache misses (compulsory, conflict and capacity) can be reduced by increasing the cache block size?
   (d) (2%) What type of cache misses can be reduced by increasing set associativity?

10. (14%) What is the average CPI for each of the following 4 schemes taking to execute the code sequence below? (Note: For the pipeline scheme, there are five stages: IF, ID, EX, MEM, and WB. We assume the reads and writes of register file can occur in the same clock cycle, and the stall circuits are available.)

```
add   $t3, $s1, $s2
sub   $t1, $s1, $s2
lw    $t2, 100($t3)
sub   $s1, $t1, $t2
```

   (a) (2%) single cycle scheme;
   (b) (4%) multi-cycle scheme without pipelining;
   (c) (4%) pipelined scheme without data forwarding hardware;
   (d) (4%) pipelined scheme with data forwarding hardware (one from EX/MEM to ALU input, and the other from MEM/WB to ALU input) available.