國 立 交 通 大 學 試 題 紙

九十六學年度第一次
博士班資格考

科目：計算機架構　　日期： 97 年 1 月 24 日　第 1 頁 共 5 頁

請"✓"明　✓不可看書　可看書

* 請將答案依題號順序寫入答冊

1. (30 points) Choose one correct answer in each question to get one point. There is a 0.3 points penalty if your answer is not correct.

( ) (1) Which one has the properties that the critical system design issues are price, power consumption, and application-specific performance? (a) desktop (b) server (c) embedded.

( ) (2) SPEC is an organization about (a) benchmark (b) embedded computers (c) IC technology.

( ) (3) The idea of making the common case fast is from (a) Moore's Law (b) Amdahl's Law (c) Rule of Thumb.

( ) (4) After improving computer organization, which one will not be affected? (a) Instruction count (b) CPI (c) Clock rate.

( ) (5) What is our goal? (a) to increase IPC (b) to increase CPI (c) to increase seconds/cycle.

( ) (6) To increase clock rate (a) will not influence cycles/instruction (b) may influence cycles/instruction (c) can not judge the influence of cycles/instruction.

( ) (7) Which one is not an embedded RISC processors? (a) ARM (b) MIPS 16 (c) SPARC.

( ) (8) Which architecture has the disadvantage of higher instruction count? (a) reg-reg (b) reg-mem (c) mem-mem.

( ) (9) What is the addressing mode of reg[r4]←reg[r4]+mem[100+reg[r1]]? (a) reg (b) immediate (c) displacement.

( ) (10) In general, what is the most frequently used instruction? (a) load (b) add (c) and (d) sub (e) store.

( ) (11) Which one does not have code compression? (a) IBM's CodePack (b) SPARC (c) ARM Thumb.

( ) (12) Graph coloring algorithm is used for (a) local optimization (b) global optimization (c) register allocation.

( ) (13) SSE (streaming SIMD extension) is belongs to (a) ARM (b) Intel (c) AMD.

( ) (14) Forward and bypassing can reduce (a) data hazard stalls (b) control hazard stalls (c) structural hazard stalls.

( ) (15) Delayed branches can reduce (a) data hazard stalls (b) control hazard stalls (c) structural hazard stalls.

( ) (16) TLP is (a) transaction-level parallelism (b) top-level parallelism (c) thread-level parallelism.

◎ 請用深黑色鋼筆或原子筆出題

命題老師簽名：

( ) (17) Which one is not the difference between Pentium III and Pentium 4? (a) deeper pipeline (b) register renaming (c) trace cache (d) micro operations.

( ) (18) The idea of window in multiple issues is about (a) set of registers (b) set of instructions (c) set of pipeline stages.

( ) (19) What is correct about the multiple instruction issue? (a) It is similar to multicycle. (b) It is not related to instruction-level parallelism. (c) It is designed for multiply instruction. (d) Its CPI can be less than one.

( ) (20) Which step will not change the status of ROB (reorder buffer)? (a) Execute (b) Write result (c) Commit.

( ) (21) To do VLIW (a) needs the help of compiler (b) does not need the help of compiler (c) does not give sufficient information to judge whether it needs the help of compiler or not.

( ) (22) Branch-target buffer stores (a) predicted address (b) prediction scheme (c) correlation bits.

( ) (23)Which one, in a pipeline, is performed in the second step? (a) instruction fetch (b) register fetch (c) data fetch (d) address computation.

( ) (24)The action MDR←Memory[ALUOut] is for (a) jump (b) branch (c) load (d) store.

( ) (25) Talking about CPI, (a) mulicycle is better than pipeline (b) pipeline is better than multicycle (c) these two are similar (d) it is not easy to justify.

( ) (26) The ideal speedup of pipeline is (a) proportional to the clock rate (b) equal to the number of instruction types (c) equal to the number of stages (d) proportional to the CPI.

( ) (27) Hardware resource conflict is a (a) structural hazard (b) branch hazard (c) data hazard (d) control hazard.

( ) (28) Data hazard can be solved by compiler through (a) data forwarding (b) dynamic scheduling (c) reordering instructions sequence.

( ) (29) The purpose of register renaming is to (a) remove antidependence (b) increase readability (c) construct name dependence (d) increase the usage of register.

( ) (30) What is correct about loop unrolling? (a) not a technique of compilers (b) a technique of instruction-level parallelism (c) not a technique for multiple instruction issue (d) a technique of branch prediction.

2. (10 points) The performance enhancement possible with a given improvement is limited by the amount that the improved feature is used. This concept is referred to as *Amdahl's law* in computing. Let's consider a general model that the subsystem-A operations responsible for a seconds and the subsystem-B operations responsible for b seconds of the total execution time, t seconds. We also

recognize that the improvement needs costs. Assume that the subsystem-A needs cost of C_A to get 10/9 improvement and it continues needing cost of C_A to get 10/9 improvement of the improved subsystem-A, i.e., 100/81 improvement of the original subsystem-A. Assume the improvement is restricted as the above discrete function. Subsystem-B follows the same rule with the discrete 10/9 improvement and discrete cost of C_B. Suppose the subsystem-A has n_A times improvements and subsystem-B has n_B times improvements. The question is that under the total cost limitation C_L, you are asked to discuss how to formulate the problem to get the maximum improvement by improving both subsystem-A and subsystem-B.

(a) Calculate both the costs to improve subsystem-A and subsystem-B. (3 points)

(b) Calculate both the responsible time of improved subsystem-A and subsystem-B. (3 points)

(c) Formulate the problem you are going to solve. (4 points)

3. (10 points) Explain true data dependence, antidependence, and output dependence and indicate every occurrence in the following example.

```
DIV.D   F0,F2,F4
ADD.D   F6,F0,F8
S.D     F6,0(R1)
SUB.D   F8,F10,F14
MUL.D   F6,F10,F8
```

part A

part B

4. (10 points) Consider the following program:

```
int i, int j, double result[4][100], double a[101][4]
for (i=0; i<4; i++)
{
   for (j=0; j<100; j=j++)
        result[i][j] = a[j][0]*a[j+1][0] + 0.5;
}
```

Arrays ***result*** and ***a*** contain 8 bytes double precision floating point elements.

- Assume the following:
- The program is running on a machine with an L1 data cache.
- The cache is fully associative with 100 blocks and an LRU replacement policy. The block size is 16 bytes. It is write-through and no write-allocate.
- Assume that only the accesses to the array locations generate loads to the data cache. The rest of the variables are all allocated in registers.
- The arrays are stored in row major form.
- The arrays start at cache line boundaries.
- Initially, the data cache is empty.

(a) [6 points] Explain which loads to the L1 data cache result in misses for the above program. Give the total number of such misses and indicate which are capacity, conflict, and compulsory (cold) misses. Assume that the processor issues loads in the order in which they appear in the program.

(b) [4 points] Now consider again the original code with the original cache hierarchy. Please rewrite the original code using loop interchange to avoid the capacity misses in the original code?

5. (20 points) For the following problem, assume an in-order 5-stage pipelined architecture that has functional units that take the following numbers as latencies:

Instruction producing result	*Instruction using result*	*Latency in clock cycles*
FP ALU op	**Another FP ALU op**	**3**
FP ALU op	**Store double**	**2**
Load double	**FP ALU op**	**1**
Integer op	**Branch**	**1**

The following code computes a dot product. Assume that `r1` and `r2` contain addresses of arrays of floating-point numbers, and `r3` contains the length of the arrays (in elements). Assume that `r4` is initialized to zero. Then, the dot product can be computed as follows:

```
loop:   ld      f5, 0(r1)      ; load element from first array
        ld      f6, 0(r2)      ; load element from second array
        multd   f7, f5, f6     ; multiply elements
        addd    f4, f4, f7     ; add elements to accumulator in f4
        addi    r1, r1, #8     ; increment pointers
        addi    r2, r2, #8
        subi    r3, r3, #1     ; decrement element count
        bnez    r3, loop       ; continue until all elements done
```

Please ignore the branch delay slot after the bnez.

(a) [3 points] How many cycles does each iteration take, without arranging the code?

(b) [4 points] What is the lowest number of cycles per iteration you can achieve by only rearranging code. (no unrolling)? Please show the scheduled code.

(c) [5 points] Unroll the given loop once, and schedule it to avoid any stalls.

(d) [4 points] How many cycles per iteration does this unrolled loop achieve?

(e) [4 points] If you were to unroll the loop 8 times, how many cycles per iteration would this achieve? (*hint: you do not need to actually perform the unrolling to answer this question)*

6. (20 points) Consider a multiprocessor system that uses broadcast snooping cache coherence. Each processor has write-back cache. Four processors, P1, P2, P3, and P4 perform the following sequence of loads and stores to/from lines A and B. Assume the protocol described in the book, which uses the three states: Invalid (I), Shared (S), and Exclusive (E). The table below shows the state of the memory system as time flows down. The cache blocks are represented with the following notation: *address:(state, data)*. For example, *A:(I,0)* means that cache block with line address A is in state I

with data value 0. A data value of x means the value is unknown or undefined. Complete the table below, updating the cache and memory states in response to the sequence of loads and stores. You may use arrows (as shown ->) to indicate that the state has not changed in that cycle.

(a) [10 points] Assume A and B belong to different blocks and do not conflict in the data caches.

P1	P2	P3	P3 P4	MEMORY
A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:0 B:0
load A A:(S, 0) B:(I, x)	->	->	->	->
->	**load B** A:(I, x) B:(S, 0)	- >	->	->
	store B = 3			
		store A = 1		
load B				
			store A = 2	

(b) [10 points] Assume A and B belong to the same cache blocks.

P1	P2	P3	P3 P4	MEMORY
A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:(I,x) B:(I,x)	A:0 B:0
load A A:(S,0) B:(S,0)	->	->	->	->
->	**load B** A:(S,0) B:(S,0)	- >	->	->
	store B = 3			
		store A = 1		
load B				
			store A = 2	

◎ 請用深黑色鋼筆或原子筆出題

命題老師簽名：