浙江大学 2017－2018 学年秋冬学期

《计算机组成与设计》课程期末考试试卷

课程号： 67190020 ，开课学院： 信息与电子工程学院

考试试卷： √A 卷、B 卷（请在选定项上打 √）

考试形式： √闭、开卷（请在选定项上打 √），

允许带 1 张 A4 大小的手写资料和计算器入场

考试日期： 2018 年 1 月 18 日，考试时间： 120 分钟

诚信考试，沉着应考，杜绝违纪。

考生姓名：_____学号：_____所属院系（专业）：_____

题序	一	二	三	四	五	六	七	八	总 分
得分									
评卷人									

I. CHOICE 1 (30 points)

1. CPU execution time for a program depends on the following **except (_____)**. A

A. Process management　　B. CPI

C. Instruction count　　D. Clock cycle time

2. In immediate addressing the operand is placed **(_____)**. A

A. After OP code in the instruction　　B. In the CPU register

C. In memory　　D. In stack

3. Which of the following program is used to translate assembly language into machine language? **(_____)** B

A. Compiler　　B. Assembler

C. Linker　　D. Loader

4. The average number of cycles taken to execute the set of instructions can be made to be less than one by following **(_____)**. C

A. ISA　　B. Pipe-lining

C. Super-scaling　　D. Sequential

5. For a virtual memory with translation look-aside buffer (TLB), which of following will be run first during memory access? **(_____)** B

A. test cache hit　　B. test TLB hit

C. test physical memory hit　　D. test dirty bit

6. The drawback of building a large memory with SRAM is **(_____)**. A

A. The large cost factor　　B. The inefficient memory organization

C. The slow speed of operation　　D. All of the mentioned

7. The temporal aspect of the locality of reference means **(_____)**. B

A. That the recently executed instruction won't be executed soon

B. That the recently executed instruction will be executed soon again

C. That the recently executed instruction is temporarily not referenced

D. None of the mentioned

8. When the process is returned after an interrupt service **(_____)** should be loaded again. D

i) Register contents ii) Condition codes

iii) Stack contents iv) Return addresses

A. i, iv B. ii, iii and iv

C. iii, iv D. i, ii

9. The write-through procedure is used **(_____).** C

A. to write onto the memory directly

B. to write and read from memory simultaneously

C. to write directly on the memory and the cache simultaneously

D. None of the mentioned

10. The frame pointer is used in MIPS to ensure that the base address, for all references to the stack frame of a function, does not change during the execution of the function. Thus, the use of the frame pointer is highly recommended in a function that **(_____)**. C

A. is a leaf function that does not call another function.

B. is a recursive function

C. makes multiple allocations in the stack

D. has lots of parameters and some of them have to be passed in the stack

11. The fastest data access is provided using **(_____)**. D

A. Caches B. DRAM's

C. SRAM's D. Registers

12. Consider a computation that consists of calculating the sum of thousands of integers. Using a simple single-cycle datapath as a starting point, what of the following techniques or approaches that *each* can NOT increase performance by a factor of four or more (that is, altogether these four techniques could result in a 256x speedup!) **(_____)**. C

A. pipelining

B. instruction-level parallelism via superscalar execution

C. multithreading

D. data-level parallelism via vector instructions

13. Which method of representation has two representations for '0'? **(_____)** A

A. Sign-magnitude B. 1's complement

C. 2's complement D. None of the mentioned

14. **(_____)** is used to implement virtual memory organization. A

A. MMU B. Frame table

C. Page table D. None of the mentioned

15. You want to maximize the performance of an application with the following characteristics: large number of data structures, low spatial locality, high temporal locality including frequent loads and stores to the same variable, 50% of the memory accesses are stores. Assuming a fixed total cache size, which set of choices below would most likely lead to increased performance. (____) A

A. Small block size, High associativity, LRU replacement, Write back, Write allocate

B. Large block size, Low associativity, LRU replacement, Write back, Write allocate

C. Small block size, High associativity, MRU replacement, Write through, No write allocate

D. Large block size, Low associativity, LRU replacement, Write through, Write allocate

II. CHOICE 2 (30 points)

1. Consider a virtual memory system with 32-bit virtual byte address, 4 KB/page, 32 bits each entry. The physical memory is 512 MB. Then, the total size of page table needs **(_____)**. C

A. 1 MB
B. about 3 MB
C. 4 MB
D. 8 MB

2. Consider two level-1 data caches; both are 2 bytes in size, 1 byte per block. Cache C1 is direct-mapped, cache C2 is 2-way set associative with least recently used (LRU) replacement. Suppose a program issues load instructions which access bytes of memory in the following sequence: 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, …. for a large number of iterations. What is the approximate hit rate for C1 and C2? **(_____)** C

A. C1: 33%, C2: 33%
B. C1: 0%, C2: 33%
C. C1: 33%, C2: 0%
D. C1: 0%, C2: 0%

3. Assume that integer is stored as a word (4 bytes) in memory, and -2 is stored at Memory address start from 0x00000004, so what Byte is stored in memory address 0x00000007 in MIPS computer? **(_____)** A

A. 1111 1110
B. 1111 1111
C. 1000 0000
D. 0000 0010

4. For a 32-bit cache-memory system, a 32 KB, 4-way set-associative cache has 2 words cache line size, how many bits are there in such cache's tag? **(_____)** A

A. 19
B. 21
C. 23
D. 25

5. Two processors A and B have clock frequencies of 700 MHz and 900 MHz respectively. Suppose A can execute an instruction with an average of 3 steps and B can execute with an average of 5 cycles. For the execution of the same instruction which processor is faster? **(___)** A

A. A
B. B
C. Both take the same time
D. Insufficient information

6. You have to write MIPS assembly code to execute the following C language statement:

A[B[j]] = x + y;

The values of A, B, j, x, and y are in memory. The minimum number of loads and stores that the

MIPS code that executes this statement must have is **(_____)**. B

A. 5 loads and 1 store
B. 6 loads and 1 store
C. 2 loads and 1 store
D. 2 loads and 2 stores

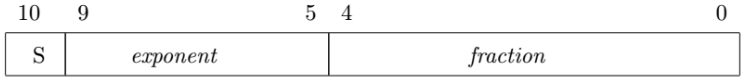
7. A processor with 32-bit addresses uses a 32 KB 2-way set associative cache with 16-byte blocks, how many bits are used for offset, index, and tag? **(_____)** A

A. 4, 10, 18
B. 8, 10, 18
C. 4, 10, 16
D. 8, 10, 16

8. *Forwarding* is a technique used in a pipeline to reduce the number of stall cycles that are caused by hazards. Each sequence of instructions shown below cause a hazard for the version of the MIPS pipeline that we studied in class. The pipeline bubble that would be caused by some of these hazards can be avoided through the use of forwarding. Others cannot. Mark the sequences for which the bubble can **NOT** be avoided through forwarding. **(_____)** A

A.
```
lw $t0, 0($t1)
add $t2, $t3, $t0
```
B.
```
sub $t0, $t1, $t2
add $t3, $t4, $t0
```
C.
```
lw $t0, 0($t1)
sll $t5, $t6, $t7
add $t2, $t3, $t0
```
D.
```
add $t7, $t8, $t9
beqz $t7, L2
```

9. The following is a format for the binary representation of a floating-point number:



The exponent is expressed in excess-8 format (also known as a bias representation). Given the binary representation above, the decimal value of the number represented can be computed by the following expression:

$$N = \begin{cases} 0.0 & \text{if } exponent = 0 \text{ and } fraction = 0 \\ (-1)^S \times 0.fraction \times 2^{-14} & \text{if } exponent = 0 \text{ and } fraction \neq 0 \\ (-1)^S \times 1.fraction \times 2^{exponent-15} & \text{if } 0 < exponent < 30 \\ (-1)^S \times \infty & \text{if } exponent = 31 \text{ and } fraction = 0 \\ NaN & \text{if } exponent = 31 \text{ and } fraction \neq 0 \end{cases}$$

Let Y = 5.2510. Give the normalized binary representation for Y. **(_____)** A

A. 0 10001 01010
B. 0 10000 01010
C. 0 10001 01011
D. 1 10000 01010

10. Consider a simple in-order five-stage pipeline with a two-cycle branch misprediction penalty and a single-cycle load-use delay penalty. For a specific program, 30% of the instructions are loads, 20% are branches, the remaining 50% of instructions are simple single-cycle ALU operations. Half of the load instructions are followed immediately by a dependent instruction, and 75% of branches are predicted correctly. What is the average CPI of this program on this processor? **(_____)** B

A. CPI = 2
B. CPI = 1.25

C. CPI = 1.5 D. CPI = 2.25

III. TRUE OR FALSE (10 points)

1. _ T _ In modern computers, CPU and Memory are connected by BUS.
2. _ T __ Von Neumann architecture has data and instructions in the same memory space.
3. _ F __ In the memory hierarchy, as the speed of operation increases the memory size also increases.
4. _ F __ In pipelining the task which requires the least time is performed first.
5. _ F __ A least recently used (LRU) replacement policy will always be better than a random replacement policy for managing virtual memory pages.
6. _ T __ Pipeline designs improve CPI over multi-cycle designs by overlapping the execution of multiple instructions.
7. _ T __ The number of pipe stages per instruction affects throughput, not latency.
8. _ T __ The average number of memory accesses per instruction is less than one if not all instructions are loads or stores.
9. _ F __ The direct memory access (DMA) is worse for large transfers than interrupts due to the overhead of setting up the transfer.
10. _ F __ Bigger cache blocks always lead to a higher hit rate.

IV. VIRTUAL MEMORY (10 points)

We have a byte-addressable toy computer that has a physical address space of 512 bytes. The computer uses a simple, one-level virtual memory system. The page table is always in physical memory. The page size is specified as 8 bytes and the virtual address space is 2 KB.

1. How many bits of each virtual address is the virtual page number? (2 points)

$$\log_2\left(\frac{virtual\ address\ space\ size}{page\ size}\right) = \log_2\left(\frac{2\ K}{8}\right) = \mathbf{8\ bits}$$

2. How many bits of each physical address is the physical frame number? (2 points)

$$\log_2\left(\frac{physical\ address\ space\ size}{page\ size}\right) = \log_2\left(\frac{512}{8}\right) = \mathbf{6\ bits}$$

We would like to add a 128-byte write-through cache to enhance the performance of this computer. However, we would like the cache access and address translation to be performed simultaneously. In other words, we would like to index our cache using a virtual address, but do the tag comparison using the physical addresses (virtually-indexed physically-tagged). The cache we would like to add is direct mapped, and has a block size of 2 bytes. The replacement policy is LRU. Answer the following questions:

3. How many bits of a virtual address are used to index into the cache? Which bits exactly? (3 points)

$$\log_2(\#blocks) = \log_2\left(\frac{cache\ size}{block\ size}\right) = \log_2(64) = \mathbf{6\ bits}$$

Bits 1..6 (Bit 0 is used for byte in block)

4. What is the size of the total tag store in bits? (3 points)

Per cache block we have tag + valid bit (no dirty bit for write-through cache).
Given that we use bits of the virtual page number to index in the cache (see v.), and we have physically tagged cache, we must use the entire physical page number as the tag.
Tag size = 6 bits (from ii) → Tag store entry size = 6 bits + 1 bit (valid bit) == 7 bits

$$Total\ tag\ store = \#blocks \cdot (tag\ store\ entry\ size) = 64 \cdot 7 = \mathbf{448\ bits = 56\ bytes}$$

V. PIPELINE (10 points)

Given the following loop expressed in a high-level language:

```
do {
  if ( VECTA[i] >= 0 ) {
    VECTB[i] = VECTA[i];
  } else {
    VECTB[i] = 0;
  }
  i++;
} while (i != N);
```

The program has been compiled in MIPS assembly code assuming that registers $t6 and $t7 have been initialized with values 0 and N respectively. The symbols VECTA, VECTB, and VECTC are 16-bit constant. The processor clock frequency is **1 GHz**.

Num.Stalls THEN	Num.Stalls ELSE	INSTRUCTION	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	Hazard Type THEN	Hazard Type ELSE
		DO: lw $t2, VECTA($t6)	IF	ID	EX	ME	WB										
		slt $t0, $t2, $0		IF	ID	EX	ME	WB									
		bne $t0, $0, ELSE			IF	ID	EX	ME	WB								
		sw $t2, VECTB($t6)				IF	ID	EX	ME	WB							
		j INC					IF	ID	EX	ME	WB						
		ELSE: sw $0, VECTB($t6)						IF	ID	EX	ME	WB					
		INC: addi $t6, $t6, 4							IF	ID	EX	ME	WB				
		bne $t6, $t7, DO								IF	ID	EX	ME	WB			

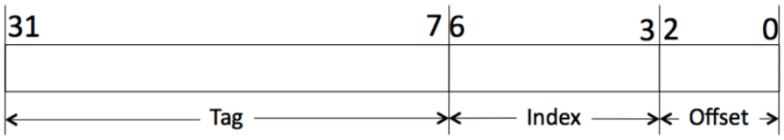
Let us consider the loop executed by 5-stage pipelined MIPS processor **without** any optimization in the pipeline **and that in the 50% of the cases (VECTA[i] >= 0)**

- Identify the **RAW (Read After Write)** data hazards by marking with (- - - - -) and control hazards with (———). (4 points)
- Identify the number of stalls to be inserted before each instruction (or between the stage IF and ID of each instruction) necessary to solve the hazards. (6 points)

Num. Stalls THEN	Num. Stalls ELSE	ISTRUZIONE	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	Hazard Type THEN	Hazard Type ELSE
3	3	DO: lw $t2,VECTA($t6)	IF	ID	EX	ME	WB									CNTR	CNTR
3	3	slt $t0,$t2,$0		IF	ID	EX	ME	WB								RAW $t2	RAW $t2
3	3	bne $t0, $0, ELSE			IF	ID	EX	ME	WB							RAW $t0	RAW $t0
3		sw $t2,VECTB($t6)				IF	ID	EX	ME	WB						CNTR (RAW $t2)	
		j INC					IF	ID	EX	ME	WB						
	3	ELSE:sw $0,VECTB($t6)						IF	ID	EX	ME	WB					CNTR
3		INC: addi $t6,$t6,4							IF	ID	EX	ME	WB			CNTR	
3	3	bne $t6,$t7, DO								IF	ID	EX	ME	WB		RAW $t6	RAW $t6

VI. CACHE (10 points)

TinyProc Inc. is a new company in the market delivering tiny processors for the toy robot market. One of its processors, TP512, has a split data and instruction level 1 cache. The data cache is two-way set associative and the instruction cache directly mapped. Both caches use the following address mapping:



(1) What is the storage capacity (expressed in bytes) of each cache? (2 points)

In both caches, each block has $2^3 = 8$ bytes.

The instruction cache (direct mapped) has $2^4 = 16$ entries, each with a single block, thus it can store $2^7 = 128$ bytes.

The data cache (2-way set associative) also has $2^4 = 16$ entries, but each entry has two blocks (one for each set), thus it can store $2^8 = 256$ bytes.

(2) All Tiny Proc Inc processors use the MIPS instruction set. A synthetic benchmark is a program that does not do any useful computation, instead it is created to enable computer architects to analyze the performance of individual components of a design such as the branch predictor or the memory hierarchy. Assume that the following synthetic benchmark, which executes an infinite loop, is executed in TP512. Execution starts at address 0x8000 0000.

```
0x8000 0000            add $v0, $zero, $zero
0x8000 0004            lui $s0, 0x4000
0x8000 0008  snippet1: lw $v0, 0($s0)
0x8000 000C            lw $t2, 4($s0)
0x8000 0010            add $v0, $v0, $t2
0x8000 0014            j snippet2
0x8000 0018            add $v0, $v0, $v0
....
0x8000 0080  snippet2: lw $t2, 128($s0)
0x8000 0084            add $v0, $v0, $t2
0x8000 0088            lw $t2, 132($s0)
0x8000 008C            add $v0, $v0, $t2
0x8000 0090            j snippet1
0x8000 0094            jr $ra
```

In the tables below, indicate the memory addresses (expressed in hexadecimal), the tag, the index, and the offset for each access performed to the instruction and to the data cache by this synthetic benchmark. You must list the accesses into the tables in the order in which they occur in the program. Under the columns First Iteration and Second Iteration indicate if the access results in a hit or a miss in that iteration (list the outcome of accesses that occur before the first iteration of the loop in the First Iteration column and use "—" if the access does not occur in the iteration). For the tag you can use a combination of hexadecimal and binary notation, such as 0xAF78 87_01 where the digits before the " _" symbol are in hexadecimal and the ones after the" _" are in binary.

There might be more lines in each of the tables than the number of accesses performed by the program in one iteration. Leave extra lines blank. (8 points)

Instruction Cache:

Memory Address	Tag	Index	Offset	First Iteration	Second Iteration

Data Cache:

Memory Address	Tag	Index	Offset	First Iteration	Second Iteration

Instruction Cache:

Memory Address	Tag	index	Offset	First Iteration	Second Iteration
0x8000 0000	0x8000 00_0	000 0	000	Miss	—
0x8000 0004	0x8000 00_0	000 0	100	Hit	—
0x8000 0008	0x8000 00_0	000 1	000	Miss	Miss
0x8000 000C	0x8000 00_0	000 1	100	Hit	Hit
0x8000 0010	0x8000 00_0	001 0	000	Miss	Miss
0x8000 0014	0x8000 00_0	001 0	100	Hit	Hit
0x8000 0080	0x8000 00_1	000 0	000	Miss	Hit
0x8000 0084	0x8000 00_1	000 0	100	Hit	Hit
0x8000 0088	0x8000 00_1	000 1	000	Miss	Miss
0x8000 008C	0x8000 00_1	000 1	100	Hit	Hit
0x8000 0090	0x8000 00_1	001 0	000	Miss	Miss

Data Cache:

Memory Address	Tag	index	Offset	First Iteration	Second Iteration
0x4000 0000	0x4000 00_0	000 0	000	Miss	Hit
0x4000 0004	0x4000 00_0	000 0	100	Hit	Hit
0x4000 0080	0x4000 00_1	000 0	000	Miss	Hit
0x4000 0084	0x4000 00_1	000 0	100	Hit	Hit