浙江大学 2018～2019 学年 春夏 学期

《 计算机组成 》课程期末考试试卷（A）

课程号： 21186031__，开课学院：**计算机学院/软件学院**

考试试卷：√A 卷、B 卷（请在选定项上打√）

考试形式： 闭 卷，允许带 一页 A4 纸手写笔记 入场，笔记署名，不得互借

交卷方式： 试卷名字朝外对折整齐，草稿纸、笔记与试卷一起上交 。

考试日期： **2019 年 06 月 18 日(10:30～12:30)** ，考试时间： **120** 分钟

I. True or False(8x1%; √/×)

Statement	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Answer								

(1). ( ) We can connect at most 7 disks to a cable of SCSI bus.

(2). ( ) There is a bit in status register, or cause register, this bit can be used to disable all kinds of hardware interrupt.

(3). ( ) In memory-mapped I/O system, dedicated I/O instruction can be used, these I/O instruction can specify both the device number and the command word.

(4). ( ) Write buffer is a small cache that can hold a few values waiting to go to main memory. It can be used to entirely eliminate any write stalls.

(5). ( ) When write-through cache scheme is used, some read misses still cause miss block in cache to be written back to main memory.

(6). ( ) C0(Coprocessor 0#) is used to process exception and float point number calculation.

(7). ( ) Floating point addition includes: [redacted], Adding Significands, Over/underflow processing, Rounding, Normalisation.

(8). ( ) Underflow means that the number is too small to be represented.

II. Choose 1 best answer. (10x2%)

(1) What is the sign extension (16 bit to 32 bit) result of number 0x89ab represented in 2's complement?

A. 0x000089ab B. 0xffff89ab C. 0x111189ab D. 0x800089ab

(2) MIPS addressing mode includes base addressing, immediate addressing, PC-relative addressing, etc, which instruction uses base addressing?

A. addi $s0,$s0,4 B. bne $s0,$s1, L1

C. j 4000 D. sw $s1,0($s0)

(3) A BNE instruction with machine code 0x1508000c is located at memory address 0x2004, suppose jump action occurs when executing this BNE instruction, the address of next instruction executed will be _____ .

A. 0x2038 B. 0x2034 C. 0x2014 D. none of above

(4) When write miss happens in a write-through cache system, the data is only written to main memory, it is not written into the cache, such cache technology is called _____.

A. Fetch-on-miss B. write back

C. write around D. none of above

[redacted]

(5) Which instruction will not cause overflow?

A. addu $t2,$t1,$t1 B. subiu $t2,$t1,0x23

C. addi $t2,$t1,0x23 D. nor $t2,$t1,$t1

(6) Which one is not a bus standard?

A.SCSI B. PCI C. DIMM D. EISA

(8) In virtual memory system, each TLB row (or called entry) contains serveral fields, which one is not such field?

A. Virtual page number

B. physical page number

C. Valid bit indicating TLB hit

D. A bit called D-bit indicating disk address is used.

(9) Which instruction can jump to an address at any position of 4GB memory?

A. jr $ra B. bne $t0,$t1,label

C. j label D. jal label

(10) Which instruction is not relative to floating point hardware?

A. bc0t B. mul.s

C. div.d D. bc1f

(7) For page table of virtual memory, what is the usage of dirty bit?
A. indicate the entry is filled with useless physical page address.
B. indicate the entry is filled with dirty physical page address.
C. indicate the TLB relative to this entry is filled with dirty data.
D. indicate new data has been written into the corresponding physical page.

III. (14%):

fill corresponding letter of A-K into table below.

Question	(1)	(2)	(3)	(4)	(5)	(6)	(7)
Answer							

(1).A number is representd as 0x7F800000 in IEEE 754 Single precision number format, its value is _____.

(2). IEEE754 single precision representation for -1 is _____.

(3). A number is represented as 10101101_00010000_00000000_00000000 in 2's complement format, original value of this number is _____.

(4). A number is represented as 0xF0000320 in 1's complement format, original value of this number is _____.

(5).Value of machine code for instruction 'jr $ra' is _____.

(6).Value of machine code for instruction 'andi $t2, $t2, 4'

(3). A number is represented as 10101101_00010000_00000000_00000000 in 2's complement format, original value of this number is _____.
(4). A number is represented as 0xF0000320 in 1's complement format, original value of this number is _____.

(5).Value of machine code for instruction 'jr $ra' is _____.

(6).Value of machine code for instruction 'andi $t2, $t2, 4'
is _____.
(7).Value of machine code for instruction 'sub $t0,$t1,$t2'
is _____.

Options for above blanks are listed below:

(A) 0x03E00008 (B) 0XBF800000 (C) 0XBF100000 (D) -0x0FFFFCDF

(E) 0x314A0004 (F) $+\infty$ (G) -0x52F00000 (H) 0x12A4022

(I) 0x12A4020 (J) -0x52700000 (K) None of above

浙江大学 2019～2020 学年 春夏 学期

《 计算机组成 》课程期末考试试卷（A）

课程号： 21186031__，开课学院：**计算机学院/软件学院**

考试试卷：√A 卷、B 卷（请在选定项上打√）

考试形式： 闭 卷，允许带 **一页 A4 纸手写笔记** 入场，**笔记署名，不得互借**

交卷方式： **试卷名字朝外对折整齐，草稿纸、笔记与试卷一起上交** 。

考试日期： **2020 年 09 月 11 日(10:30～12:30)** ，考试时间： **120** 分钟

OPcode: R-Type:0, Beq:4, Bne:5, J:2, Lw:35 SW:43

Function Code: Add:32, Sub:34, And:36, or:37, jr:8,

Register No. $s0:16, $t0:8

1. (20%)

1.1 (4%) 1) Assume that registers $s0 and $s1 hold the values 0x80000000 and 0xD0000000, respectively.

1) What is the value of $t0 for the following assembly code

```
add $t0,$s0,$1
```

A: 0x50000000 (overflow)

2) What if the value $t0 for the following assembly code

```
add $t0, $s0, $s1
add $t0, $t0, $s0
srl $t0,$t0,  2
```

A: 0x34000000

1.2 (2%) Assume the current PC is 0x1FFFFFFC, what's next value of PC after execution of "j 0x10"?

A: 0x10000100 ({PC[31:28],0x10(26 bits),00})

1.3 (2%)What is the hexadecimal value of 2020.

A: 0X7E4

1.4 (4%)Show the IEEE754 for the floating-point number **-32.6** and **0.64** in single precision. Write down the binary representation.

-32.6

1	10000100	00000100110011001100100

0.64

0	011 1111 110	10100011110101110000101

1.5(4%) For a processor with the clock rate 2.5Ghz, the instructions can be divided into four classes (Class A, B, C, D) according to their CPIs of 1, 2, 3,3. Given a program with instruction count of 1.0E6 instructions divided into classes as follows: 10% class A, 20% class B, 50% Class C, and 20% class D.

a. What is the global CPI for the program? 2.6;

b. The clock cycles required for executing the program? 2.6 $x10^6$;

1.6 (4%) Convert the MIPS instructions into machine code or Machine code into MIPS instructions.

Address(Hex)	MIPS Assembly Instruction	Machine Code(Hex)
100000	Loop: Beq $s0, $t8, L1	(0x12180030)
100004	(sub $s0, $s1, $s2)	0x02328022
100008	J Loop	(0x804000)
……		-
1000D4	L1: ……	-

2. (20%)Memory Hierarchy

2.1(10%) For a direct-mapped cache design with a 32-bit address, the following bits of the address are used to access the cache.

Tag	Index	Offset
31-11	10-6	5-0

1) What is the cache block size(in words)? 16 (1 分)

2) How many entries does the cache have? 32 (1 分)

3) The following byte-addressed cache reference are recorded.

0	4	16	132	160	1024	30	140	3000

How many blocks are replaced? 0 (2 分)

What is the hit ratio 5/9 (55.6%) (2 分)

4) List the final state of the cache after 3), with each valid entry represented as a record of <index, tag, data> (每个 Cache 状态 1 分)。
[0,0,0~63]。
[2,0,128~181]。
[16,0,1024~1087]。
[14,1,2944~3007]。
其他位置均为无效。

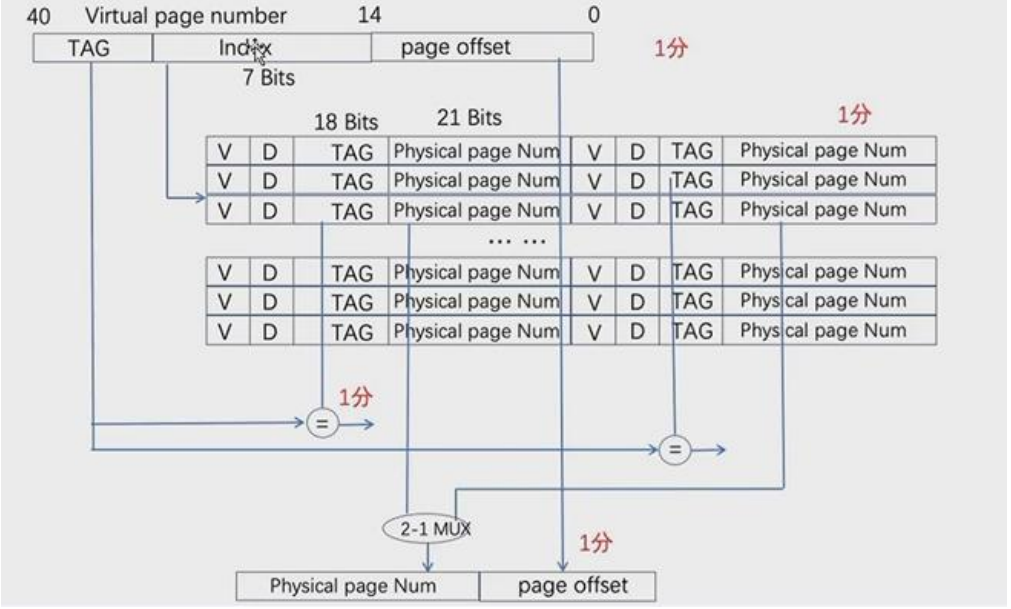
2.2(10%)Consider a virtual memory system with the following properties:
- 40-bit virtual byte address;
- Page size 32KB;
- 36-bit physical byte address.

1) What is the total size of the page table for each process on this processor? Assume that the valid and dirty take a total of 2 bits and that all the virtual pages are in use and the disk address are not stored in the page table. Each Entry of the page table should be byte aligned.

A: 1) Entry number: $2^{40}/32KB = 2^{25}$ (2 分)。
2) Entry size: 36 bits - 15 bits + 2 bits = 23 bits.
Byte aligned: 3 Bytes. (2 分)。
3) 3 Bytes x 2^{25} = 3 x 32M = 96MB; (2 分)。

2)Assume the virtual memory system is implemented with a two-way set associative TLB with 256 TLB entries. Show the virtual-to-physical mapping with a figure.

1) virtual memory address 分成 三部分： 1 分；
2) 2-way TLB 结构 ： 1 分；
3) index 过程： 1 分；
4) physical Address 过程： 1 分；



3(30%) Program

3.1(10%)

```
START: addi $s0, $zero, $zero
       addi  $s1, $zero, $zero
       addi  $s2, $zero, $zero
       addi  $t2, $zero,3
       la    $t1, LOOP1 #Load the addr of the Loop1 into $t1
       lw    $t3,4($t1)
       addi  $t3,$t3,4
       sw    $t3,4($t1)
       lw    $t3,8($t1)
       addi  $t3,$t3,8
       sw     $t3,8($t1)
       lw     $t3,($t1)

LOOP1:
       bne  $s1, $s2, LOOP2
       addi  $s0, 252.
```

```
       addi  $s0, 120
LOOP2:
       addi  $s0, 100
       addi  $s0, 50
       addi  $s1, 1
       addi  $t3,1
       sw    $t3,0($t1)
       subi $t2, 1
bne    $t2, $zero, LOOP1
END:   … …
```

After the instructions running, the content of register $s0 is (584(0x248)), Why? (Give all the different values of register $s0 during the program running)

1)First Loop: $s0=0; $s0=$s0+256; $s0=$s0+128; $s0=$s0+100; $s0=$s0+50;

2) Second Loop: $s0= $s0+50;

3) Thirt Loop: $s0 保持不变;

答案：4 分； 每个 LOOP 过程$s0 的变化： 2 分；

3.2 (20%) Implement the following C code in MIPS, assuming that the sort is the first function called:

```
void sort (int v[],int n){
    int i, j;
    for(i=1;i<n;i++){
       for(j=i-1;j>=0;j--){
          if(compare(v[j],v[j+1])){
             swap(v,j)
          }
       }
    }
}
int compare(int a, int b){
     if( a >= b)
        return 1;
     else
        return 0;
}
int swap(int v[], int k){
      int temp;
      temp = v[k];
      v[k] =v[k+1];
      v[k+1] =v[k];
}
```

Be sure to handle the stack and frame pointer appropriately. i corresponds to $s0, j corresponds to $s1 in function **sort**, temp correspond to $s2 in function **swap**.

1) Write the MIPS assembly code corresponding to this C function.

2) Draw the status of the stack before calling sort and during each function call. Indicate the names of registers and variables stored on the stack and mark the location of $sp and $fp.

```
sort:addi $sp, -20;
     sw  $ra, 16($sp)
     sw  $s3, 12 ($sp)
     sw  $s2, 8 ($sp)
     sw  $s1, 4($sp)
     sw  $s0,  0($sp)
(以上 Stack 保护 1 分，$s0,$s1 必须保护，其他根据代码情况)
     add $s2,$a0,$zero;//s1=V[]
     add $s3,$a1,$zero;//13=n
(以上参数传递 1 分)
     add $s0,$zero  //i=0
f1st: slt $t0,$s0,$s3
     beq  $t0, $zero exit1//i<n
     addi  $s1,-1;//j=i-1;
(第一个循环 1 分)
f2st:  slti $t0, $s1,0; //$j<0;
     bne   $t0,zero, exit2;
     sll   $t0,$s1,2// $t1=j*4;
     add   $t2, $s2,$t1 //v+j*4;
     lw     $a0, 0($t2) //v[j];
     lw     $a1, 4($t2) //v[j+4];
     jal    compare;
(准备 V[j],V[j+1]，正常跳转 3 分)
     beq    $v0, zero, exit2
     add    $a0, $a0, $s2;
     add    $a1, $a0, $$1;
     jal    swap
     addi   $s1,$s1,-1
     j      f2st.
(正常传递 swap 参数，跳转 2 分)
exit2:  addi.  $s0, $s0,-1;
     j      f1st
```

```
void sort (int v[],int n){
    int i, j;
    for(i=1;i<n;i++){
       for(j=i-1;j>=0;j--){
if(compare(v[j],v[j+1])){
            swap(v,j)
         }
       }
    }
}
```

	(正常判断结束 1 分)
	exit1: lw $s0, 0($sp)
	lw $s1, 4($sp)
	lw $s2, 8($sp)
	lw $s3, 12($sp)
	lw $ra, 16($sp)
	addi $sp, $sp,20
	jr $ra
	(正常返回 2 分)

int compare(int a, int b){
if(sub(a,b) >= 0)
return 1;
else
return 0;
}
	compare:
	slt $t0,$a0, $a1;//a<b
	beq $t0,zero, exit3;
	addi $v0, $zero,1
	exit3:
	addi $v0, $zero,$zero;
	jr $ra
	(compare 函数 2 分)

int swap(int v[], int k){
int temp;
temp = v[k];
v[k] =v[k+1];
v[k+1] =v[k];
}
	swap:
	sll $t1,$a1,2 //$t1=k*4;
	add $t1, $a0,$t1 //v+k*4
	lw $t0, 0($t1)
	lw $t2, 4($t1)
	sw $t2, 0($t1)
	sw $t0, 4($t1)
	jr. ra
	(swap 函数 2 分)

4.(20%)Multicycle CPU Design

1)(6%) Exception detection is an important aspect of exception handling. Try to identify the cycle in which the following exception can be detected for the multicycle DataPath. The steps of multicycle CPU is shown below.

Step name	Action for R-type instructions	Action for memory-reference instructions	Action for branches	Action for jumps
Instruction fetch	IR <= Memory[PC] PC <= PC + 4			
Instruction decode/register fetch	A <= Reg [IR[25:21]] B <= Reg [IR[20:16]] 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) PC <= ALUOut	PC <= (PC [31:28], (IR[25:0]),2'b00))
Memory access or R-type completion	Reg [IR[15:11]] <= ALUOut	Load: MDR <= Memory[ALUOut] or Store: Memory [ALUOut] <= B		
Memory read completion		Load: Reg[IR[20:16]] <= MDR		

a. Overflow exception (EX 执行完检测，有学生说在 MEM 阶段也可以，因为 EX 阶段得到结果，在 MEM 检测；)

b. Divide by zero exception. (Assume we use the ALU for division in on cycle) (EX，这个在计算前就知道，所以一定是 EX 阶段；)

c. Invalid instruction (ID)

d. External interrupt (Every stage is OK，我们设计中可以规定只在 IF 阶段或者结束执行检测)

e. Invalid instruction memory address (IF)

f. Invalid data memory address (EX 或者 MEM 阶段，看学生解释，因为 EX 阶段地址算出来了， 而 MEM 使用地址，都是合理的；)