King Fahd University of Petroleum and Minerals
College of Computer Science and Engineering
Computer Engineering Department

ICS 233: COMPUTER ARCHITECTURE & ASSEMBLY LANGUAGE

Term 142 (Spring 2014-2015)
Major Exam 1
Saturday February 28, 2015

Time: 120 minutes, Total Pages: 9

Name:_____ ID:_____ Section: _____

Notes:

- Do not open the exam book until instructed
- Answer all questions
- All steps must be shown
- Any assumptions made must be clearly stated

Question	Max Points	Score
Q1	25	
Q2	15	
Q3	20	
Total	60	

Dr. Aiman El-Maleh
Dr. Mayez Al-Muhammad

[25 Points]

(Q1) Fill in the blank in each of the following questions:

(1) Assuming 8-bit 2`s complement representation, the hexadecimal number EA is equal to the decimal number -22.

(2) Two advantages of programming in assembly language are accessibility to hardware resources and space and time efficiency.

(3) Assuming that variable Array is defined as shown below:

Array: .byte 1, 2, 3, 4, -1, -2, -3, -4

After executing the following sequence of instructions, the content of the three registers is $t1=0xffffffff $t2=0x0000feff and $t3=0xfcfdfeff.

```
la $t0, Array
lb $t1, 4($t0)
lhu $t2, 4($t0)
lw $t3, 4($t0)
```

(4) Given a magnetic disk with the following properties:

- Rotation speed = 7200 RPM (rotations per minute)
- Average seek = 8 ms, Sector = 512 bytes, Track = 200 sectors

The average time to access a block of 100 consecutive sectors is 16.33 ms.

Average access time = Seek Time + Rotation Latency + Transfer Time
Rotation time in milliseconds = 1000*60/7200 = 8.33 ms
Rotation Latency = 8.33/2 = 4.167 ms
Time to transfer 100 sectors = (100/200)* 8.33 = 4.167 ms
Average access time = 8 + 4.167 + 4.167 = 16.33 ms.

(5) Assuming the following data segment, and assuming that the first variable X is given the address **0x10010000**, then the addresses for variables Y and Z will be 0x10010004 and 0x1001000c**.**

```
.data
X:    .byte  1, 2, 3
Y:    .half  4, 5, 6
Z:    .word 7, 8, 9
```

(6) Assume that the instruction j NEXT is at address 0x00400010 in the text segment, and the label NEXT is at address 0x00400fec. Then, the address stored in the assembled instruction for the label NEXT is 0x00400fec/4=0x01003fb.

(7) Assume that the instruction bne $t0, $t1, NEXT is at address 0x00400010 in the text segment, and the label NEXT is at address 0x00400fec. Then, the address stored in the assembled instruction for the label NEXT is (0x00400fec-0x00400014)/4=0xfd8/*4*=0x03f6

(8) The pseudo instruction `bge $s2, 5, Next` is implemented by the following minimum MIPS instructions:

```
slti $at, $s2, 5
beq $at, $0, Next
```

(9) The pseudo instruction *ori $t0, 0x12345678* is implemented by the following minimum MIPS instructions:

```
lui $at, 0x1234
ori $at, $at, 0x5678
or $t0, $t0, $at
```

(10) After executing the instruction addu $t0, $s1, $s2, the following MIPS instruction can be used to store the carry out of addition in register $t1:

```
sltu $t0, $t0, $s1
```

Check again.

(11) To multiply the **signed** content of register $t0 by 62.5 without using multiplications and division instructions, we use the following MIPS instructions:

```
sll $t1, $t0, 6
sra $t2, $t0, 1
subu $t1, $t1, $t2
subu $t0, $t1, $t0
```

62.5= 64-2+1/2 ; last term is missing !

(12) The content of register **$s1** after executing the following code is 0x4.

```
        li $s0, 0x5a
        li $s1, 0
Next:
        andi $t0, $s0, 1
        add $s1, $s1, $t0
        srl  $s0, $s0, 1
        bne $s0, $0, Next
```

[15 Points]

(Q2) Answer the following questions.

(i) For what reasons there is a bandwidth mismatch between the speed of processor and the speed of main-memory. How this mismatch problem can be alleviated.

DRAM access time is many folds that of the processor clock and this will increase over time due to quite different growth rates between DRAM and Processor. The mismatch can be alleviated by fetching on demand blocks of data/code (containing the requested item) and storing in a cache memory, where the access time is close to that of the CPU.

(ii) Since year 2000, the clock frequency of Intel IA32 processors was growing at a rate of approximately 54% per year. In 2000, the reference clock rate was 1.25 GHz. What was the expected clock frequency of IA32 processors in year 2005.

Clock rate in year 2005: 1.25GHz x (1.54)^5= 10.83 Ghz

(iii) Consider a binary number A=[a_{31},….., a_0]. Express A as a weighted sum of its components (a_i) and their corresponding power of 2 assuming A is (1) unsinged, and (2) signed (2'scomplement). Determine the least and largest values for each of the above two cases.

$A = 2^{31}\ a_{31} + 2^{30}\ a_{30} + \ldots + 2^{0}\ a_0$ *Its range is* $[\ 0\ ,\ 2^{32} - 1\ ]$

$A = -2^{31}\ a_{31} + 2^{30}\ a_{30} + \ldots + 2^{0}\ a_0$ *Its range is* $[\ -2^{31}\ ,\ 2^{31} - 1\ ]$

(iv) Evaluate the decimal value corresponding to the 32-bit signed 2's complement binary number:

1101 0000 0111 1100 1111 1111 1100 0000

1098 7654 3210 9876 5432 1098 7654 3210
1101 0000 0111 1100 1111 1111 1100 0000

$N = -2^{31} + 2^{30} + 2^{28} + (2^{23} - 2^{18}) + (2^{16} - 2^{6}) = -797114432$

(v) Translate to MIPS assembly the following statement A[15] = A[10] + A[5]+ 5. Assume register $s0 is used to store the base address of array A[], which is an array of words.

```
lw $t0, 20 ($s0)
lw $t1, 40 ($s0)
add $t0, $t1, $t0
addi  $t0, $t0, 5
sw    $t0, 60 ($s0)
```

[20 Points]

(Q3) Write separate MIPS assembly code fragments with **minimum** instructions to implement each of the given requirements.

(i) Given an array of integers (i.e. words), Array, with its base address stored in register $a0 and its size stored in register $a1, write a MIPS assembly program to store the smallest integer in Array into register $v0.

```
        lw      $v0, ($a0)
        addi    $a1, $a1, -1
loop:   addi    $a0, $a0, 4
        lw      $t0, ($a0)
        bge     $t0, $v0, skip
        move    $v0, $t0
skip:   addi    $a1, $a1, -1
        bne     $a1, $0, loop
```

(ii) Suppose $s1 originally contains 0x12abcd34. Write a MIPS assembly program to extract the 2nd to 5th hexadecimal digits from $s1 and place the extracted data in the upper part of register $s2 and the lower part is filled with zeros.

```
srl $s1, $s1, 4
sll $s1, $s1, 16
```

(iii) Write the most optimized MIPS assembly program to implement the following compound expression:

if ([($s1 > 0) && ($s2 < 0)] || ($s3 > 0)) {$s4++;}

```
        bgtz    $s3, Then
        blez    $s1, Endif
        bgez    $s2, Endif
Then:   addiu   $s4, $s4,1
Endif:
```

(iv) Write the most optimized MIPS assembly program for the following WHILE statement:

```
i = 0;
WHILE (a[i]+b[i] <= k) i = i+1;
```

Assume $s0 and $s1 are used to store the base addresses of arrays a[] and b[], respectively. $s2 and $s3 are used to store the index i and the value k, respectively. Assume that a[] and b[] are arrays of words.

```
        xor     $s2, $s2, $s2
        j       cond
loop:   addiu   $s2, $s2, 1
        addiu   $s0, $s0, 4
        addiu   $s1, $s1, 4
cond:   lw      $t0, 0($s0)
        lw      $t1, 0($s1)
        addu    $t0, $t0, $t1
        ble     $t0, $s3, loop
```

MIPS Instructions:

Instruction	Meaning	R-Type Format					
add $s1, $s2, $s3	$s1 = $s2 + $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x20
addu $s1, $s2, $s3	$s1 = $s2 + $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x21
sub $s1, $s2, $s3	$s1 = $s2 – $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x22
subu $s1, $s2, $s3	$s1 = $s2 – $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x23

Instruction	Meaning	R-Type Format					
and $s1, $s2, $s3	$s1 = $s2 & $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x24
or $s1, $s2, $s3	$s1 = $s2 | $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x25
xor $s1, $s2, $s3	$s1 = $s2 ^ $s3	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x26
nor $s1, $s2, $s3	$s1 = ~($s2|$s3)	op = 0	rs = $s2	rt = $s3	rd = $s1	sa = 0	f = 0x27

Instruction	Meaning	R-Type Format					
sll $s1,$s2,10	$s1 = $s2 << 10	op = 0	rs = 0	rt = $s2	rd = $s1	sa = 10	f = 0
srl $s1,$s2,10	$s1 = $s2>>>10	op = 0	rs = 0	rt = $s2	rd = $s1	sa = 10	f = 2
sra $s1, $s2, 10	$s1 = $s2 >> 10	op = 0	rs = 0	rt = $s2	rd = $s1	sa = 10	f = 3
sllv $s1,$s2,$s3	$s1 = $s2 << $s3	op = 0	rs = $s3	rt = $s2	rd = $s1	sa = 0	f = 4
srlv $s1,$s2,$s3	$s1 = $s2>>>$s3	op = 0	rs = $s3	rt = $s2	rd = $s1	sa = 0	f = 6
srav $s1,$s2,$s3	$s1 = $s2 >> $s3	op = 0	rs = $s3	rt = $s2	rd = $s1	sa = 0	f = 7

Instruction	Meaning	I-Type Format			
addi $s1, $s2, 10	$s1 = $s2 + 10	op = 0x8	rs = $s2	rt = $s1	imm^{16} = 10
addiu $s1, $s2, 10	$s1 = $s2 + 10	op = 0x9	rs = $s2	rt = $s1	imm^{16} = 10
andi $s1, $s2, 10	$s1 = $s2 & 10	op = 0xc	rs = $s2	rt = $s1	imm^{16} = 10
ori $s1, $s2, 10	$s1 = $s2 | 10	op = 0xd	rs = $s2	rt = $s1	imm^{16} = 10
xori $s1, $s2, 10	$s1 = $s2 ^ 10	op = 0xe	rs = $s2	rt = $s1	imm^{16} = 10
lui $s1, 10	$s1 = 10 << 16	op = 0xf	0	rt = $s1	imm^{16} = 10

Instruction	Meaning	Format			
j label	jump to label	op^6 = 2	imm^{26}		
beq rs, rt, label	branch if (rs == rt)	op^6 = 4	rs^5	rt^5	imm^{16}
bne rs, rt, label	branch if (rs != rt)	op^6 = 5	rs^5	rt^5	imm^{16}
blez rs, label	branch if (rs<=0)	op^6 = 6	rs^5	0	imm^{16}
bgtz rs, label	branch if (rs > 0)	op^6 = 7	rs^5	0	imm^{16}
bltz rs, label	branch if (rs < 0)	op^6 = 1	rs^5	0	imm^{16}
bgez rs, label	branch if (rs>=0)	op^6 = 1	rs^5	1	imm^{16}

Instruction	Meaning	Format					
slt rd, rs, rt	rd=(rs<rt?1:0)	op^6 = 0	rs^5	rt^5	rd^5	0	0x2a
sltu rd, rs, rt	rd=(rs<rt?1:0)	op^6 = 0	rs^5	rt^5	rd^5	0	0x2b
slti rt, rs, imm^{16}	rt=(rs<imm?1:0)	0xa	rs^5	rt^5	imm^{16}		
sltiu rt, rs, imm^{16}	rt=(rs<imm?1:0)	0xb	rs^5	rt^5	imm^{16}		

Instruction	Meaning	I-Type Format			
lb rt, imm^{16}(rs)	rt = MEM[rs+imm^{16}]	0x20	rs^5	rt^5	imm^{16}
lh rt, imm^{16}(rs)	rt = MEM[rs+imm^{16}]	0x21	rs^5	rt^5	imm^{16}
lw rt, imm^{16}(rs)	rt = MEM[rs+imm^{16}]	0x23	rs^5	rt^5	imm^{16}
lbu rt, imm^{16}(rs)	rt = MEM[rs+imm^{16}]	0x24	rs^5	rt^5	imm^{16}
lhu rt, imm^{16}(rs)	rt = MEM[rs+imm^{16}]	0x25	rs^5	rt^5	imm^{16}
sb rt, imm^{16}(rs)	MEM[rs+imm^{16}] = rt	0x28	rs^5	rt^5	imm^{16}
sh rt, imm^{16}(rs)	MEM[rs+imm^{16}] = rt	0x29	rs^5	rt^5	imm^{16}
sw rt, imm^{16}(rs)	MEM[rs+imm^{16}] = rt	0x2b	rs^5	rt^5	imm^{16}