ECE 2035 Programming HW/SW Systems Spring 2017

4 problems, 6 pages Exam One Solutions 8 February 2017

Problem 1 (2 parts, 30 points) **Loops**

Part A (12 points) Given an array `int A[100]` of **non-unique** integers **sorted** in increasing order, complete the following code fragment which computes the mode of the data in A. Note that each element in the array is a member of a run whose length is greater than or equal to one.

```
int   A[100]   =   {-42,22,22,75,75,75...121};   //   given
int i;
int ThisNum,Mode;
int ThisNumCnt,ModeCnt;

Mode=ThisNum=A[0];
ModeCnt=ThisNumCnt=0;

for (i=0;i<100;i++){
     if (ThisNum == A[i])
          ThisNumCnt++; // increment run length of ThisNum
     else {

           ThisNum = A[i];                ; //A.1

           ThisNumCnt = 1;                ;
     }
     if (ThisNumCnt >= ModeCnt) {

           Mode = ThisNum;                ; //A.2

           ModeCnt = ThisNumCnt;          ;

     }
}
printf("The mode is %d with %d occurrences\n",Mode,ModeCnt);
```

A.1 Update ThisNum and ThisNumCnt for the next distinct value in the array A.

A.2 Update Mode and ModeCnt.

Part B (18 points) Write MIPS code for the for loop fragment in Part A. Assume ThisNum, Mode, ThisNumCnt and ModeCnt are in registers $4, $5, $6, and $7 respectively. Also assume that the address of array A is in $10. Your loop control variable should be in $1. You may assume all variables are initialized as given. *For maximum credit use a minimum number of instructions.*

Label	Instruction	Comment
Mode:	lw $4, 0($10)	# ThisNum = A[0]
	lw $5, 0($10)	# Mode = A[0]
	addi $6, $0, 0	# ThisNumCnt = 0
	addi $7, $0, 0	# ModeCnt = 0
	addi $1, $0, 0	# initial i = 0
Loop:	slti $8, $1, 400	# is i below loop limit?
	beq $8, $0, Exit	# if not, exit loop
	add $11, $10, $1	# Array base + offset
	lw $9, 0($11)	# A[i]
	bne $9, $4, Else	# If ThisNum != A[i], do Else.
	addi $6, $6 1	# If ThisNum == A[i], ThisNumCnt++
	j If2	# jump to 2nd if.
	# part A.1	#
Else:	addi $4, $9, 0	# Else: ThisNum = A[i]
	addi $6, $0, 1	# ThisNum = 1
If2:	slt $8, $6, $7	# if ThisNumCnt < ModeCnt
	bne $8, $0, Skip	# skip part A.2
	# part A.2	#
	addi $5, $9, 0	# Mode = ThisNum
	addi $7, $6, 0	# ModeCnt = ThisNumCnt
Skip:	addi $1, $1 4	# i++
	j Loop	# loop back
Exit:	…	# code after the loop…

Problem 2 (2 parts, 20 points) **Conditionals: Nested if-then-else**

Consider the following MIPS code:

```
.data
ALoc: .word 10
BLoc: .word 20
CLoc: .word 30
DLoc: .word 30
Result: .alloc 1
.text
       lw   $1, ALoc($0)
       lw   $2, BLoc($0)
       lw   $3, CLoc($0)
       lw   $4, DLoc($0)
       ori  $5, $0, 1
       slt  $6, $3, $4
       bne  $6, $0, Write
       beq  $1, $2, Write
       addi $5, $0, 0
Write: sw   $5, Result($0)
       jr   $31
```

Part A (8 points) What is written to the memory location labeled Result by this program?

Result: _**0**_

Part B (12 points) Suppose the labels ALoc, BLoc, CLoc, DLoc, and Result are the addresses of the C variables A, B, C, D, and R. What C expression is being computed by this code? Write your C-code in terms of A, B, C, and D and use logical predicates.

R = **(C < D) || (A == B)**;

Problem 3 (4 parts, 25 points) **Assembly Programming**

Part A (6 points) Suppose L1 = 0x00234500 in the original code below. Write the **sequence** of MIPS instructions that is necessary to achieve the intent of the original instruction.

Original:	Equivalent MIPS instructions:
`addi $3, $0, L1`	**`lui $3, 0x0023`**
	`ori $3, $3, 0x4500`

Alternative solution:

```
lui $3, 0x2345
sll $3, $3, 8
```

Part B (6 points) Write a **single** MIPS instruction that is equivalent to the original fragment. Assume big *endian* byte ordering.

Original:	Equivalent MIPS statement:
`lui  $4, 0xFF00`	**`lb $3, 1000($0)`**
`lw   $3, 1000($0)`	
`and  $3, $3, $4`	
`sra  $3, $3, 24`	

Part C (7 points) Write a MIPS fragment with 1 **instruction** that is equivalent to the original fragment under a certain condition.

Original:	Equivalent MIPS in **one** instructions only:
`      bne $1, $2, next`	**`beq $1, $2, Target`**
`      j Target`	
`next:`	

Under what condition are the fragments equivalent? **The address labeled by Target is an offset from the branch instruction address that can fit in 16 bits.**

Part D (6 points) What hexadecimal value will be in register $2 when this MIPS fragment executes? Assume big *endian* byte ordering.

```
.data
In: .word 0xABCD1234
.text
    addi $3, $0, In
    lbu  $2, 2($3)    # note this is lbu
```

$2: 12

Problem 4 (2 parts, 25 points) Assembly Programming

Part A (12 points) What does the following code fragment print?

```
        int i, x;
        int A[10] = {99, 33, 44, 22, 56, 78, 1, 5, 9, 88};

        for(i=0; i<10; i=i+2){
          if(i & 2)              // bitwise AND
             continue;           // and don't print x
          x = A[i];
          printf("x = %d\n", x);
        }
```

```
x = 99
x = 56
x = 9
```

Fill in the blanks to rewrite the code above to produce the equivalent behavior without using `continue`.

```
        int i;
        int A[] = {99, 33, 44, 22, 56, 78, 1, 5, 9, 88};

        for( i=0 ; i<10 ; i+=4 ){
          x = A[i];
          printf("x = %d\n", x);
        }
```

Part B (13 points) Answer the three questions below about the following C fragment.

```
   int i, j, k, count;
   j = k = count = 0;

   for (i=0; i < 9; i++){                        // outer loop
      if (i % 3) continue;
      j = 0;
      while (j < 9){                             // middle loop
        k = 0;
        while (k < 9){                           // inner loop
          if (k % 3) break;
          k++;
        }
        count++;
        j++;
      }
   }
   printf("%d\n", count);
```

How many times is `break` executed?	**27**
How many times is `continue` executed?	**6**
What is printed?	**27**