ECE 2035	Programming HW/SW Systems	Spring 2015
4 problems, 6 pages	Exam One	4 February 2015

Your Name (please print clearly) ____

This exam will be conducted according to the Georgia Tech Honor Code. I pledge to neither give nor receive unauthorized assistance on this exam and to abide by all provisions of the Honor Code.

Signed ____

1	2	3	4	total
30	20	25	25	100

Instructions: This is a closed book, closed note exam. Calculators are not permitted.

If you have a question, raise your hand; do not leave your seat.

Please work the exam in pencil and do not separate the pages of the exam.

For maximum credit, show your work.

Good Luck!

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

Part A (10 points) Write a **for loop** in C that finds the maximum element of vector A, which contains 100 integers, and assigns the maximum to the variable max. Do **not** use a break or continue statement. *For maximum credit, declare and initialize variables as needed.*

```
int A[100] = {4, -1, 3, …, 17};
int max;
```

Part B (20 points) Write a MIPS code fragment that is equivalent to the code you wrote in Part A. Write the maximum value to the memory location labeled M. *For maximum credit, include comments and use a minimal number of instructions.*

Label	Instruction	Comment
	.data	
A:	.word 4, -1, 3, …, 6, 17	# int A[100]={4,-1,3,…, 6, 17};
M:	.alloc 1	# max value will be stored here
	.text	
		# initialize variables
		# check loop exit test
		# read in current element
		# compare to maximum
		# update variables
		# continue looping
Exit:		# store maximum

Problem 2 (2 parts, 20 points) **Conditionals: Compound Predicates**

Part A (8 points) Consider the following MIPS code fragment. The comment indicates which variable each register holds. These variables are of type `int` and are initialized elsewhere.

Label	Instruction	Comment
		`# $2: I, $3: C, $9: Count, $8: temp`
	`slt  $8, $3, $0`	
	`bne  $8, $0, Next`	
	`slti $8, $3, 26`	
	`beq  $8, $0, Next`	
	`addi $9, $9, 1`	
`Next:`	`addi $2, $2, 1`	

What is the equivalent C code fragment? For maximum credit, use a compound logical predicate wherever possible.

Part B (12 points) Turn this C code fragement into the equivalent MIPS code. Assume $1 holds A, $2 holds B, $3 holds C and $4 holds D. *For maximum credit, include comments and use a minimal number of instructions.*

```
if (A && B)
  C = C | D;
else
  C = C & D;
D = C * 8;
```

Label	Instruction	Comment

Problem 3 (5 parts, 25 points) **MIPS Equivalences**

For each of the following MIPS code fragments, write a single MIPS instruction that is equivalent to the fragment.

Part A (5 points)

Original:	Equivalent MIPS statement:
`addi $1, $0, 128`	
`div $3, $1`	
`mflo $4`	

Part B (5 points)

Original:	Equivalent MIPS statement:
`lw   $5, 0($2)`	
`andi $5, $5, 0xFF00`	
`srl  $5, $5, 8`	

Part C (5 points)

Original:	Equivalent MIPS statement:
`xor $6, $1, $2`	
`beq $6, $0, Target`	

Part D (5 points)

Original:	Equivalent MIPS statement:
`addi $7, $0, 400`	
`lw   $8, 0($7)`	

Part E (5 points)

Original:	Equivalent MIPS statement:
`addi $9, $0, 0xAAAA`	
`sll  $9, $9, 16`	

Problem 4 (4 parts, 25 points) **Control Flow**

Part A (9 points) What does the following code fragment print?

```
int A[10] = {20, 40, 60, 80, 100, 120, 140, 160, 180, 200};
int B[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
int i=-1;
while(i<10){
  i++;
  if (i == 3) continue;
  if (i == 5) break;
  B[9-i] = A[i];
  }
printf("i = %d\n", i);
printf(“B[2] = %d\n", B[2]);
printf(“B[4] = %d\n”, B[4]);
printf(“B[6] = %d\n", B[6]);
printf(“B[8] = %d\n", B[8]);
```

i =

B[2] =

B[4] =

B[6] =

B[8] =

Part B (6 points) Write a MIPS code fragment that branches to label “Target” when register $1 is less than or equal to register $2. You may use only **two** instructions. You may use additional registers as needed. *For maximum credit, include comments.*

Label	Instruction	Comment

Part C (4 points) Fill in the blank in the code below to make the code fragment print 4.

```
int i, a=64;
for(i = 0; i<____; i++)
     a = a >> 2;
printf("%d", a);
```

Part D (6 points) Suppose the instruction ”`jal Foo`” is at instruction memory address 5040 and `Foo` is a label of an instruction at memory address 3024. When this instruction is executed, what changes occur to the registers. List all registers that are changed (both general purpose and special purpose) and give their new values.

Register	New Value

MIPS Instruction Set (core)

instruction	***example***	***meaning***
arithmetic		
add	add $1,$2,$3	$1 = $2 + $3
subtract	sub $1,$2,$3	$1 = $2 - $3
add immediate	addi $1,$2,100	$1 = $2 + 100
add unsigned	addu $1,$2,$3	$1 = $2 + $3
subtract unsigned	subu $1,$2,$3	$1 = $2 - $3
add immediate unsigned	addiu $1,$2,100	$1 = $2 + 100
set if less than	slt $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if less than immediate	slti $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
set if less than unsigned	sltu $1, $2, $3	if ($2 < $3), $1 = 1 else $1 = 0
set if < immediate unsigned	sltui $1, $2, 100	if ($2 < 100), $1 = 1 else $1 = 0
multiply	mult $2,$3	Hi, Lo = $2 * $3, 64-bit signed product
multiply unsigned	multu $2,$3	Hi, Lo = $2 * $3, 64-bit unsigned product
divide	div $2,$3	Lo = $2 / $3, Hi = $2 mod $3
divide unsigned	divu $2,$3	Lo = $2 / $3, Hi = $2 mod $3, unsigned
transfer		
move from Hi	mfhi $1	$1 = Hi
move from Lo	mflo $1	$1 = Lo
load upper immediate	lui $1,100	$1 = 100 x 2^16
logic		
and	and $1,$2,$3	$1 = $2 & $3
or	or $1,$2,$3	$1 = $2 | $3
and immediate	andi $1,$2,100	$1 = $2 & 100
or immediate	ori $1,$2,100	$1 = $2 | 100
nor	nor $1,$2,$3	$1 = not($2 | $3)
xor	xor $1, $2, $3	$1 = $2 ⊕ $3
xor immediate	xori $1, $2, 255	$1 = $2 ⊕ 255
shift		
shift left logical	sll $1,$2,5	$1 = $2 << 5 (logical)
shift left logical variable	sllv $1,$2,$3	$1 = $2 << $3 (logical), variable shift amt
shift right logical	srl $1,$2,5	$1 = $2 >> 5 (logical)
shift right logical variable	srlv $1,$2,$3	$1 = $2 >> $3 (logical), variable shift amt
shift right arithmetic	sra $1,$2,5	$1 = $2 >> 5 (arithmetic)
shift right arithmetic variable	srav $1,$2,$3	$1 = $2 >> $3 (arithmetic), variable shift amt
memory		
load word	lw $1, 1000($2)	$1 = memory [$2+1000]
store word	sw $1, 1000($2)	memory [$2+1000] = $1
load byte	lb $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
load byte unsigned	lbu $1, 1002($2)	$1 = memory[$2+1002] in least sig. byte
store byte	sb $1, 1002($2)	memory[$2+1002] = $1 (byte modified only)
branch		
branch if equal	beq $1,$2,100	if ($1 = $2), PC = PC + 4 + (100*4)
branch if not equal	bne $1,$2,100	if ($1 ≠ $2), PC = PC + 4 + (100*4)
jump		
jump	j 10000	PC = 10000*4
jump register	jr $31	PC = $31
jump and link	jal 10000	$31 = PC + 4; PC = 10000*4