ECE 2035	Programming HW/SW Systems	Fall 2017
3 problems, 6 pages	Exam One	22 September 2017

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	total
30	35	35	100

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

Read each question over before you start to work.

If you have a question, raise your hand; do not leave your seat. The meaning of each question should be clear, but if something does not make any sense to you, please ask for clarification.

Please work the exam in pencil and do not separate the pages of the exam. If you run out of room, please continue on the back of the previous page.

For maximum credit, show your work.

Good Luck!

Problem 1 (4 parts, 30 points) **Loops**

Consider the following C code fragment:

```
int H[100] = {1997, 2, -7, 1, 2010, 4, 3, 6, …, 17};
int i, x, y, z;
for (i = 1; i<100; i = i+4){
  x = H[i];
  y = H[i+1];
  <code block A>
 }
z = i;
```

The body of this fragment contains `<code block A>` to indicate additional instructions not shown. This block uses `i`, but does not change the value of `i`. It also does not contain *continue* or *break* statements.

Part A (4 points) How many times is `<code block A>` executed? Answer:___________.

Part B (4 points) What is the minimum value of *i*? Answer:_____________.

Part C (4 points) What is the final value of z? Answer:_____________.

Part D (18 points) Write a MIPS code fragment that is equivalent to the C code above. **Use the following register assignments: $1: i, $2: x, $3: y, $4:z**. Use additional registers if necessary. Use "<code block A>" in your MIPS code to indicate where the instructions for this code block go. *For maximum credit, include comments.* (Note: there are more blank lines provided than you need.)

Label	Instruction	Comment
	.data	
H:	**.word 1997, 2, -7, 1, 2010, …, 17**	**# int H[100]={1997, 2, …, 17};**
	.text	

Problem 2 (2 parts, 35 points) **Conditionals: Compound Predicates**

Consider the following C code fragment, where the variables `Hs, He, Ss, and Se` are integers:

```
if ((Se >= Hs) && (He >= Ss))
  {
    <code block A>
  }
else
  {
    <code block B>
  }
<code block C>
```

Part A (15 points) Write the equivalent C fragment using a logical *or* (`||`) instead of logical *and* (&&). Hint: Use DeMorgan's Theorem and swap the then and else clauses. It may be helpful to draw the control flow graph.

Part B (20 points) Turn the C code fragment into the equivalent MIPS code. **The variables are held in these registers: $1: Hs, $2: He, $3: Ss and $4: Se**. Use additional registers if necessary. Use "*<code block A/B/C>*" to indicate where the instructions for these code blocks go. *For maximum credit, include comments and use a minimal number of instructions.* (More blank lines are provided than you need.)

Label	Instruction	Comment

Problem 3 (4 parts, 35 points) **Understanding Code**

Part A (9 points) What values are in registers $1 and $2 after this MIPS code fragment executes? Express your answers in hexadecimal.

```
lui  $1, 0x1
addi $2, $1, 0xABCD
```

$1: 0x
$2: 0x

Part B (9 points) Given the following MIPS code:

```
.data
Input: .word 0xAABBCCDD

.text
       addi $3, $0, Input
```

Write a single MIPS instruction that is equivalent to the original MIPS fragment. Assume *little endian* byte ordering.

Original:	Equivalent MIPS instruction:
`addi $4, $0, 8`	
`lw   $5, 0($3)`	
`srlv $5, $5, $4`	
`andi $5, $5, 0xFF`	

Part C (8 points) What are the values of the variables x, y, z, and w after the following C code fragment executes? Express your answers in decimal. Hint: remember how C implements compound predicates.

```
int x, y, z;
x = y = z = 33;
int w = 10;
if ((x == 44) || (y = 5) || (z = 8))  // note "==" vs "="
  w = 77;
```

Variable:	Value:
x	
y	
z	
w	

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

```
int V[] = {1, 5, 7, 6, -9, 17, -20, 0, -3};
int j, i=0;
while(V[i] != 0)
  {
    printf("V[%d]: %d\n", i,V[i]);
    if (V[i] < 0)
      {
        for (j=0; j<i; j++)
          {
            if (j == 2)
              continue;
            printf("j: %d\n", j);
          }
        break;
      }
    i++;
  }
```

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 + sign_extend(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 + zero_extend(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 & zero_extend(100)
or immediate	ori $1,$2,100	$1 = $2 | zero_extend(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