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 = A[0];
int i;
for (i=1; i<100; i++)
  if (max < A[i])
    max = A[i];
```

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`	
	`addi $1, 0, 4`	`# initialize i = 1`
	`lw $2, A($0)`	`# initialize max=A[0]`
`Loop:`	`slti $3, $1, 400`	`# check loop exit test`
	`beq $3, $0, Exit`	`# if i>=100, exit`
	`lw $4, A($1)`	`# read in current element`
	`slt $3, $2, $4`	`# is max < A[i]?`
	`beq $3, $0, Skip`	`# if not, skip update`
	`addi $2, $4, 0`	`# else new max = A[i]`
`Skip:`	`addi $1, $1, 4`	`# update i`
	`j Loop`	`# continue looping`
`Exit:`	`sw $2, M($0)`	`# 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.

```
if ((c >= 0) && (c < 26))
   count++;
i++;
```

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
	`beq $1, $0, Else`	**`# if !A, branch to Else`**
	`beq $2, $0, Else`	**`# else if !B, branch to Else`**
	`or $3, $3, $4`	**`# else (if A&&B), C=C|D`**
	`j End`	**`# jump over Else`**
`Else:`	**`and $3, $3, $4`**	**`# if !A||!B, C=C&D`**
`End:`	**`sll $4, $3, 3`**	**`# D = C*8`**

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`	**`sra $4, $3, 7`**
`div $3, $1`	
`mflo $4`	

Part B (5 points)

Original:	Equivalent MIPS statement:
`lw $5, 0($2)`	**`lbu $5, 1($2)`**
`andi $5, $5, 0xFF00`	
`srl $5, $5, 8`	

Part C (5 points)

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

Part D (5 points)

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

Part E (5 points)

Original:	Equivalent MIPS statement:
`addi $9, $0, 0xAAAA`	**`lui $9, 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 = 5
B[2] = 0
B[4] = 0
B[6] = 0
B[8] = 40

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
	slt $3, $2, $1 **beq $3, $0, Target**	**# is $2<$1?** **# if not, $2<=$1, branch to Target**

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<__2__; 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
$31	**5044**
PC	**3024**