This is a closed book, closed note test. Calculators are not permitted. Please work the exam in pencil and do not separate the pages. For maximum credit, show your work.

Name:

Q1: /30

Q2: /25

Q3: /45

Total: /100
Question 1 (30 points)

Part A: (5 points) write a MIPS code fragment that branches to label “Target” when register $1$ is equal to $0x123456$. You are only allowed to use the instructions in the attached sheet. You may use at most 5 instructions. Pay attention to the number of bytes in $0x123456$. For maximum credits, include comments.

```
ADDI $2, $0, 0x12
SLL  $2, $2, 16
ADDI $2, $2, 0x3456
BEQ  $1, $2, Target
```

Part B: (10 points) Write a C code fragment that reverses the elements in a 1024-element array $A$ and stores the result in array $B$, i.e. $B[0]$ gets the value of $A[1023]$, $B[1]$ gets the value of $A[1022]$ ... $B[1023]$ gets the value of $A[0]$. You need to use a loop structure, using 1024 lines of assignment will get 0 credit. For maximum credit, declare and initialize variables as needed.

```c
int A[1024] = {2, 3, 2, 0, -4, ... };
int B[1024];

int i;
for(i=0;i<1024;i++) {
    B[i] = A[1023-i];
}
```

Part C: (15 points) Write a MIPS code fragment that is equivalent to your code in Part B.

```
.data
ArrayA: .word 2, 3, 2, 0, -4, ...
ArrayB: .alloc 1024

.text
ADDI $1, $0, $0
ADDI $2, $0, 4092
Loop:  SLTI $3, $1, 4096
       BEQ $3, $0, Exit
       LW $4, ArrayA($2)
       SW $4, ArrayB($1)
       ADDI $2, $2, -4
       ADDI $1, $1, 4
       J Loop
Exit:  Nop
```
Question 2: (25 points)

Part A: (5 points) Given the following multi-dimensional array defined in a C code fragment:

```c
int A[4][5];
```

Suppose the array starts at base address 1000. What is the address of element A[i][j]?

\[ 1000 + (i \times 5 + j) \times 4 \]

Part B: (10 points) Suppose the following variables are declared in a C program. Assuming the variables are allocated in the order that they are declared. The allocation scheme tries not to leave gaps in the memory unless necessary.

```c
int A; 1000 - 1003
double C; 1024-1031 (have to skip 1020 as it is not aligned)
char D; 1032 (will only occupy 1 byte)
int E; 1036
```

Suppose A is allocated to address 1000. List where the other variables (arrays) are allocated. For array B, you need to give the range of byte addresses that B spans over. Pay attention to the size of the data types and memory alignment.

B: 1004 - 1019
C: 1024 - 1031
D: 1032 - 1032
E: 1036 - 1039

Part C: (10 points) There are four variables A, B, C, and X. All are 32-bit integers.
A is at memory address 1000
B is at memory address 2000
C is in register $3
X's address is stored in $4

Write a MIPS code segment that computes \( X = A + B + C \). You can use any register that you deem necessary, but you cannot overwrite $3 or $4.

```
LW $1, 1000($0)
LW $2, 2000($0)
ADD $5, $1, $2
ADD $5, $5, $3
SW $5, 0($4)
```
Question 3: (45 points)

Part A: (5 points) Given the following C code segment:

```c
int i, j, sum;
sum = 0;
for(i=0;i<10;i++) {
    for(j=0;j<10;j++) {
        if (i+j > 10 ) break;
        sum ++;
    }
}
```

What is the value of sum when the above code segment completes?

for the i=0 iteration, the inner loop will not break; sum = sum + 10
for the i=1 iteration, the inner loop will not break; sum = sum + 10
for the i=2 iteration, the inner loop will break when j=9; sum = sum + 9
for the i=3 iteration, the inner loop will break when j=8; sum = sum + 8
...
for the i=9 iteration, the inner loop will break when j=2; sum = sum + 2

So in the end, sum = 10 + 10 + 9 + 8 + 7 + ... + 2 = 64

Part B: (15 points) Convert the above code segment to MIPS. Pay attention to where the break statement would jump to. Specify your register assignment (to i, j, sum).

For maximum credit, include comments.

```mips
i: $1   j: $2,   sum $3
ADDI $3, $3, 0   # sum = 0
ADDI $1, $0, 0   # i=0

ILOOP: SLTI $4, $1, 10
BEQ $4, $0, ExitI # i<10? If no, exit
ADDI $2, $0, 0   # j=0

JLOOP: SLTI $4, $2, 10
BEQ $4, $0, ExitJ # j<10? If no, exit
ADD $5, $1, $2   # $5 = i + j
SUBI $5, $5, 10  # $5 = i + j - 10
SLT $6, $0, $5   # $6 = 1 if 0<i+j-10
BNE $6, $0, ExitJ # if 10 < i + j, break out of Jloop
ADDI $3, $3, 1   # Sum ++
ADDI $2, $2, 1
J JLoop

ExitJ ADDI $1, $1, 1
J ILOOP

ExitI: NOP
```

4/6
Part C (7 points): Given the following C code segment,

```c
int a, x, y, z;
x = y = z = 0;

if ( ( (x == 1) & (y == 2) ) || ( z == 1 ) )
a = 0;
else
  a = 1;
```

1. What's the value of a after the code executes? (3 points)

**X=1 evaluates to 1, Y=2 evaluates to 2, 1 & 2 = 0, so next Z==1 will be evaluated (and it's 0). So the condition failed. So we have a = 1 afterwards.**

2. The compound condition of the if statement has 3 components (on x, y, and z). Which one(s) of the 3 will be evaluated when the above code executes? (4 points)

All three will be evaluated.

Part D (6 points): Convert the following C code segment from using “switch” to using “if ... else if ... else” and “go to”.

```c
int x, y;
switch (x) {
  case 0:
    case 1: y +=1 ; break;
    case 2: y -=2;
    case 3: y *= 3;
    case 4: y /= 4; break;
  default: y = x;
}

y ++
```

```c
if(x==0) goto Entry1;
else if (x==1) goto Entry 1;
else if (x==2) goto Entry 2;
else if(x==3) goto Entry 3;
else if (x==4) goto Entry 4;
else goto EntryD;
Entry1: y += 1; goto Exit;
Entry2: y -= 2;
Entry3: y *=3;
Entry4: y /= 4; goto Exit;
EntryD: y = x;
Exit: y ++;
```
Part E (6 points): convert the following C code segment from using while loop to using for loop.

```c
int i, sum;
sum=0;
for(i=0;i<=100;i++) {
    sum += i;
}
```

Part F (6 points): Given the following C program,

```c
int count;

int foo (int x) {
    int count;
    x = x + 1;
    count = 2;
    return x;
}

int main() {
    int y;
    y = count = 0;
    foo(y);
    printf("%d %d", y, count);
}
```

What will main() print?

```
y is local to main(), so its value will not be changed by calling foo();
count is global, so foo() might be able to change it. However, foo() declared a local variable also named count, so foo actually works on a local variable 'count'. In the end, foo will not change the global count.
```

Therefore, main() will print 0 0