1. What would be the output of the following MIPS code?

```
.globl main
main:
        addu    $s7, $0, $ra

        add     $s3, $0, $0

        addi    $s4, $0, 1
        add     $s5, $0, $0
        la      $s6, save

        .data
        .align  2
        .globl  save

# the next line creates an array of 10 words that can be referred to
as “save”
# the array is initialized to the 10 values after .word
# so the first array entry is a 0 and the last entry is a 2
save:   .word  0, 0, 0, 0, 0, 0, 0, 6, 3, 2

        .text

Loop:
        add     $t8, $s3, $s3
        add     $t8, $t8, $t8
        add     $t8, $t8, $s6
        lw      $t9, 0($t8)
        bne     $t9, $s5, Exit

        add     $s3, $s3, $s4
        j       Loop
Exit:

        .data
        .globl  message1
message1:       .asciiz "\nThe value of i is: "
        .text
        li      $v0, 4
        la      $a0, message1
        syscall
        li      $v0, 1
        add     $a0, $0, $s3
        syscall

        addu    $ra, $0, $s7
        jr      $ra
        add     $0, $0, $0
```

2. The following code fragment computes the Kronecker product of two matrices – don't worry if you have never heard of Kronecker – this problem assumes no foreknowledge of this, as you can discern what is being done by examining the code. The matrices *a* and *b* are the input matrices – and *c* is the output matrix – note that *c* is larger:

```
#define SIZE 40
#define KSIZE SIZE*SIZE

int a[SIZE][SIZE];
int b[SIZE][SIZE];
int c[KSIZE][KSIZE];
```

The actual product can be computed as follows (with somewhat inefficient code):

```
 for (i=0; i<SIZE; i++)
   for (j=0; j<SIZE; j++)
     for (x=0; x<SIZE; x++)
       for (y=0; y<SIZE; y++)
         c[i*SIZE+x][j*SIZE+y]=a[i][j]*b[x][y];
```

The choice of loop ordering is, as always, important here for memory performance. In this case, there are four nested loops – and this code is written in such a way that there are no dependencies to impede reordering. The code is shown with ordering *ijxy* (i.e. the ordering of the loops from inner to outer – as labeled by the iterator of the loop). Which of the following orderings should be the ***worst*** for this loop? By worst, we mean the ordering which will result in the ***longest*** execution time.

a. ijxy
b. jixy
c. xiyj
d. ixjy
e. jyix