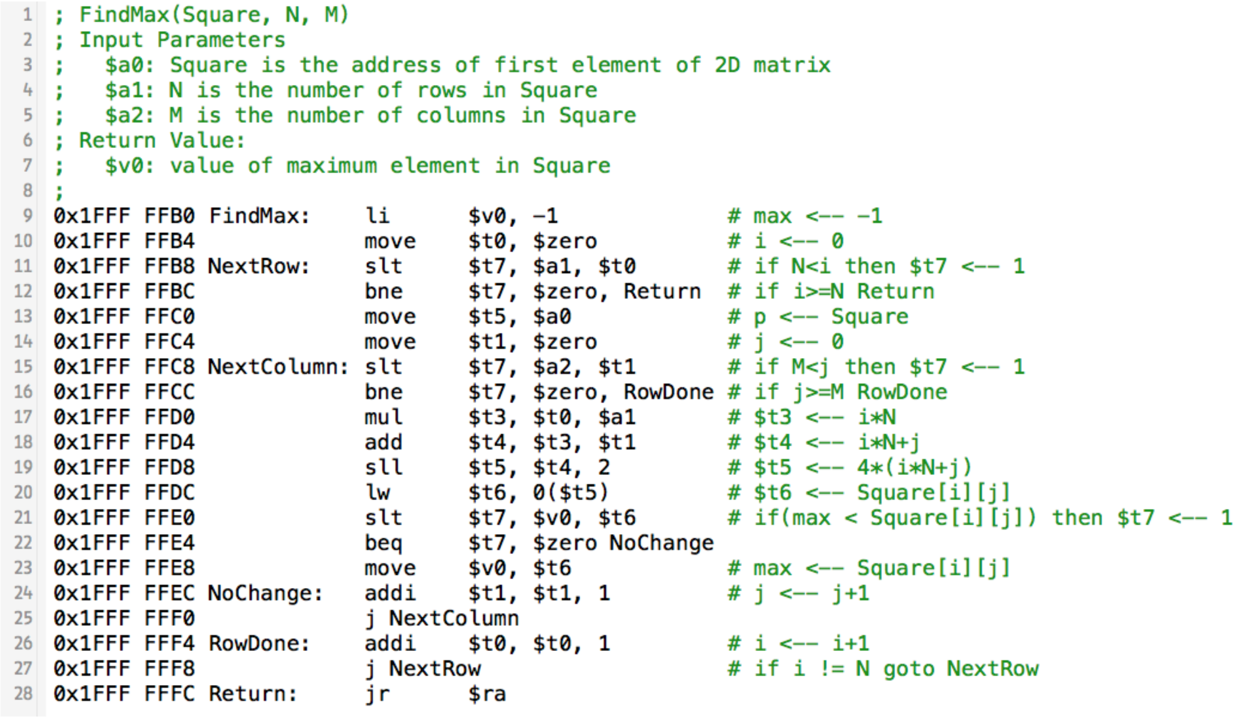

```
; FindMax(Square, N, M)
; Input Parameters
;   $a0: Square is the address of first element of 2D matrix
;   $a1: N is the number of rows in Square
;   $a2: M is the number of columns in Square
; Return Value:
;   $v0: value of maximum element in Square
;
0x1FFF FFB0 FindMax:    li      $v0, -1            # max <-- -1
0x1FFF FFB4             move    $t0, $zero         # i <-- 0
0x1FFF FFB8 NextRow:    slt     $t7, $a1, $t0      # if N<i then $t7 <-- 1
0x1FFF FFBC             bne     $t7, $zero, Return # if i>=N Return
0x1FFF FFC0             move    $t5, $a0           # p <-- Square
0x1FFF FFC4             move    $t1, $zero         # j <-- 0
0x1FFF FFC8 NextColumn: slt     $t7, $a2, $t1      # if M<j then $t7 <-- 1
0x1FFF FFCC             bne     $t7, $zero, RowDone # if j>=M RowDone
0x1FFF FFD0             mul     $t3, $t0, $a1      # $t3 <-- i*N
0x1FFF FFD4             add     $t4, $t3, $t1      # $t4 <-- i*N+j
0x1FFF FFD8             sll     $t5, $t4, 2        # $t5 <-- 4*(i*N+j)
0x1FFF FFDC             lw      $t6, 0($t5)        # $t6 <-- Square[i][j]
0x1FFF FFE0             slt     $t7, $v0, $t6      # if(max < Square[i][j]) then $t7 <-- 1
0x1FFF FFE4             beq     $t7, $zero NoChange
0x1FFF FFE8             move    $v0, $t6           # max <-- Square[i][j]
0x1FFF FFEC NoChange:   addi    $t1, $t1, 1        # j <-- j+1
0x1FFF FFF0             j NextColumn
0x1FFF FFF4 RowDone:    addi    $t0, $t0, 1        # i <-- i+1
0x1FFF FFF8             j NextRow                  # if i != N goto NextRow
0x1FFF FFFC Return:     jr      $ra
```

Figure 1: MIPS Assembly code for FindMax procedure.

This question studies the MIPS assembly code for the FindMax procedure shown in Figure 1.

Question 1 (20 points):

1. (**4 points**) Consider the following invocation of the procedure FindMax

```
lui     $a0, 0x0002
li      $a1, 0x01F4
li      $a2, 0x03E8
jal     FindMax
```

What are the values, expressed in decimal, of the parameters `N` and `M` for this call to `FindMax`?

We simply have to convert the hexadecimal values given into decimal

`N = 0x01F4` $= 16^2 + 15 \times 16 + 4 = 256 + 240 + 4 = 500$

`M = 0x03E8`$= 3 * 16^2 + 14 * 16 + 8 = 768 + 224 + 8 = 1000$

2. (**4 points**) In a given invocation of `FindMax`, `N = 10000` and `M = 5000` and the condition for the branch in line 22 is true 50% of the time. How many instructions are executed by this call?

   To solve this question, we need to analyze the assembly code to determine how many times each instruction is executed:

   - Instructions in lines 9, 10, and 28 are not inside any loop and therefore each is executed once.
   - Instructions in lines 11, 12, 13, 14, 26, and 27 are executed by the outer loop but are not executed by the inner loop. Thus each of these instructions is executed N times.
   - Instructions in lines 15, 16, 17, 18, 19, 20, 21, 22, 24, and 25 are executed once for each iteration of the inner loop. Therefore these instructions are executed $N \times M$ times.
   - Once the last iteration of the inner loop executes the jump instruction at line 25, instructions at lines 15 and 16 are executed one more time. This happens N times.
   - Similarly, when the last time that the jump instruction at line 27 is executed, the instructions at lines 11 and 12 are executed to get out of the outer loop. Thus there are two more instructions executed.
   - The instruction in line 23 is only executed when the branch in line 22 is not taken, therefore it is executed 50% of the times that the inner loop is executed. Thus, this instruction is executed $0.5 \times N \times M$.

   The number of instructions executed by `FindMax`, for this call, is given by:

   $$\begin{aligned} \# \text{ of instructions} &= 5 + N \times (8 + 10.5 \times M) \\ &= 5 + 8 \times N + 10.5 \times N \times M \end{aligned}$$

   Thus, for the specific execution, we have

   $$\begin{aligned} \# \text{ of instructions} &= 5 + 8 \times 10000 + 10.5 \times 10000 \times 5000 \\ &= 525,080,005 \end{aligned}$$

3. (**4 points**) Several executions of programs that are similar to `FindMax` have been used to determine the number of clock cycles executed by each type of instructions in the MIPS processor that is executing `FindMax`. It was determined that the following instructions take one cycle each: `li`, `move`, `slt`, `add`, `sll`, `addi`. The `mult` instruction takes five cycles. Branch

instructions take four cycles each, the jump instructions take two cycles each, and a load-word instruction takes ten cycles. How many clock cycles are necessary to execute an invocation of `FindMax` with `N = 10000` and `M = 5000` described above?

The code for `FindMax` executes double-nested loop. The outmost loop starts in line 11 and the jump instruction that returns to the start of the loop is at line 26. A similar reasoning as explained in the answer of the item above.

The number of cycles required to execute `FindMax` is:

Lines	Cycles	# Times Executed	Total Cycles
9, 10	$1+1$	1	2
11, 12	$1+4$	$N+1$	$5N+5$
13, 14, 26, 27	$1+1+1+2$	N	$5N$
15, 16	$1+4$	$N(M+1)$	$5NM+5N$
17, 18, 19, 20, 21, 22, 24, 25	$5+1+1+10+1+4+1+2$	NM	$25NM$
23	1	$0.5NM$	$0.5NM$
28	2	1	2

$$\#\text{ of clock cycles} \quad = \quad 9+15N+30.5NM$$

For the specific invocation:

$$\begin{aligned} \#ofclockcycles &= 9+15\times 10000+30.5\times 10000\times 5000 \\ &= 1,525,150,009 \end{aligned}$$

4. (**4 points**) What is the average number of clocks per instruction (CPI) for the invocation of `FindMax` with `N = 10000` and `M = 5000` described above?

$$\text{CPI} \quad = \quad \frac{\text{Number of Clock Cycles}}{\text{Number of Instructions}} = \frac{1,525,150,009}{525,080,005} = 2.9\frac{\text{Clock Cycles}}{\text{Instruction}}$$

5. **4 points**) If the invocation of `FindMax` with `N = 10000` and `M = 5000` described above is executing in a MIPS processor running with a clock frequency of 4 GHz, how long does it take to execute `FindMax`?

$$\text{Clock Cycle} = \frac{1}{4\times 10^9 Hz} = 0.25\times 10^{-9}s = 0.25\ ns$$

$$\text{Time} = \text{Number of Clock Cycles} \times 0.25\times 10^{-9}s = 1.525\times 10^9 \times 0.25\times 10^{-9} = 0.38\ s$$