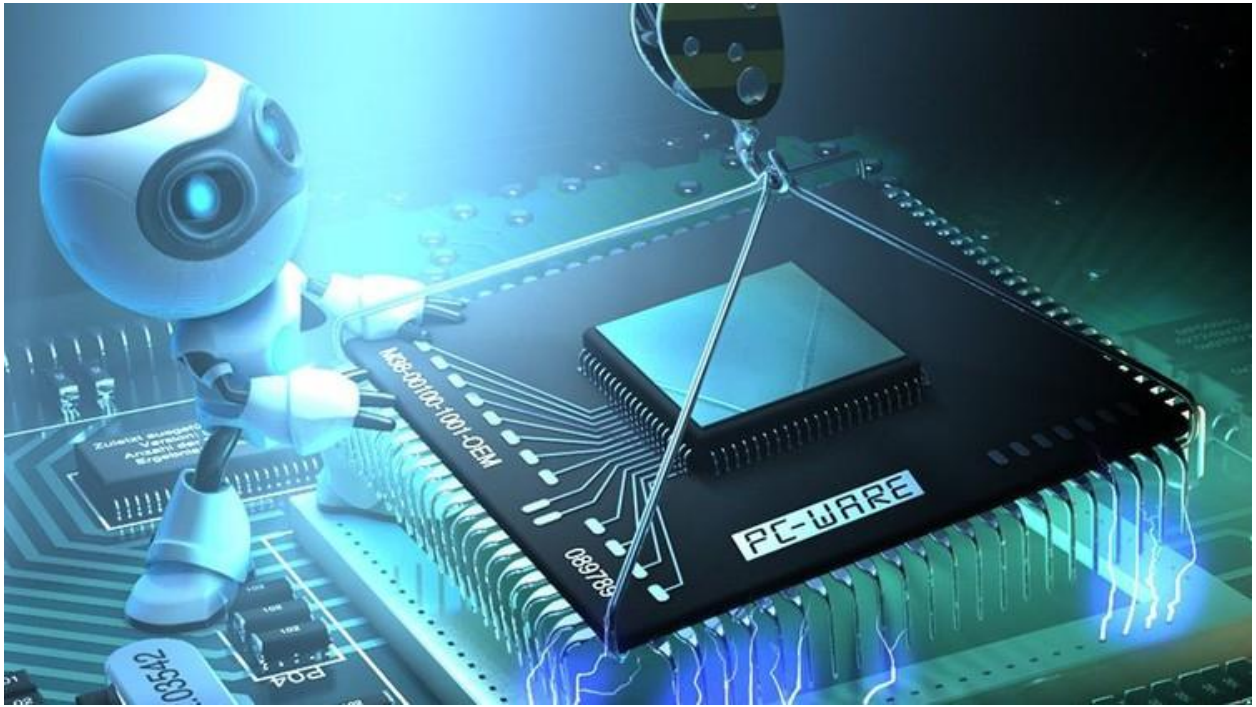
Exam 7 - Problem 2



Exam 7 - Problem 2

Given the following code written in C:

```
int Exam(int M[50][75], int n)
{ int x, y, elem, average;
 average = 0;
 elem = 0;
 for (x=0; x<50; x++)
    for (y=0; y<75; y++)
        if (M[x][y] > n) {
            average = average + M[x][y];
            elem++;
        }
 if (elem > 0)
    average = average / elem;
 return average;
}
```

Lucas Bazilio - Udemy

Exam 7 - Problem 2

Bearing in mind that the function parameters are accessible at the following addresses: address of **`M`** in **`8(%ebp)`** and **`n`** in **`12(%ebp)`**, translate the body of the subroutine into the x86 assembler. Use registers for the variables `x`, `y`, `elem`, `average`.

Exam 7 - Problem 2

```
Exam:
   pushl %ebp
   movl %esp, %ebp

   movl 8(%ebp), %edi          ; Load the address of M into %edi
   movl 12(%ebp), %ebx         ; Load n into %ebx
   movl $0, %eax               ; Initialize average to 0
   movl $0, %ecx               ; Initialize elem to 0
    movl $0, %esi              ; Initialize x to 0
```

x86

Part 1/5

Exam 7 - Problem 2

```
outer_loop_start:
    cmpl $50, %esi              ; Compare x with 50
    jge outer_loop_end          ; If x >= 50, exit the outer loop

    movl $0, %edx               ; Initialize y to 0
 inner_loop_start:
    cmpl $75, %edx              ; Compare y with 75
    jge inner_loop_end          ; If y >= 75, exit the inner loop
```

x86

Part 2/5

Exam 7 - Problem 2

```
; Calculate the memory address for M[x][y]
        movl %esi, %eax         ; Copy x into %eax
        imull $75, %eax         ; Multiply x by 75
        addl %edx, %eax         ; Add y to %eax
        imull $4, %eax          ; Multiply the sum by 4

        addl %eax, %edi         ; Add the offset to the base address of M
        movl (%edi), %edx       ; Load M[x][y] into %eax
```

Part 3/5

x86

Exam 7 - Problem 2

```
    cmpl %ebx, %edx              ; Compare M[x][y] with n
    jle inner_loop_increment     ; If M[x][y] <= n, skip incrementing
    addl %edx, %eax              ; Add M[x][y] to average
    incl %ecx                    ; Increment elem
 inner_loop_increment:
    incl %edx                    ; Increment y
    jmp inner_loop_start         ; Continue the inner loop

inner_loop_end:
   incl %esi                    ; Increment x
   jmp outer_loop_start         ; Continue the outer loop
```

x86

Part 4/5

Exam 7 - Problem 2

```
outer_loop_end:
    cmpl $0, %ecx                    ; Compare elem with 0
    jle skip_average_calculation     ; If elem <= 0, skip average calculation

    cltd                    ; Sign-extend %eax into %edx (required for idivl)
    idivl %ecx              ; Divide average by elem

    skip_average_calculation:
    movl %eax, -4(%ebp)             ; Move the final average into -4(%ebp)

    popl %ebp
    ret
```

x86

Part 5/5