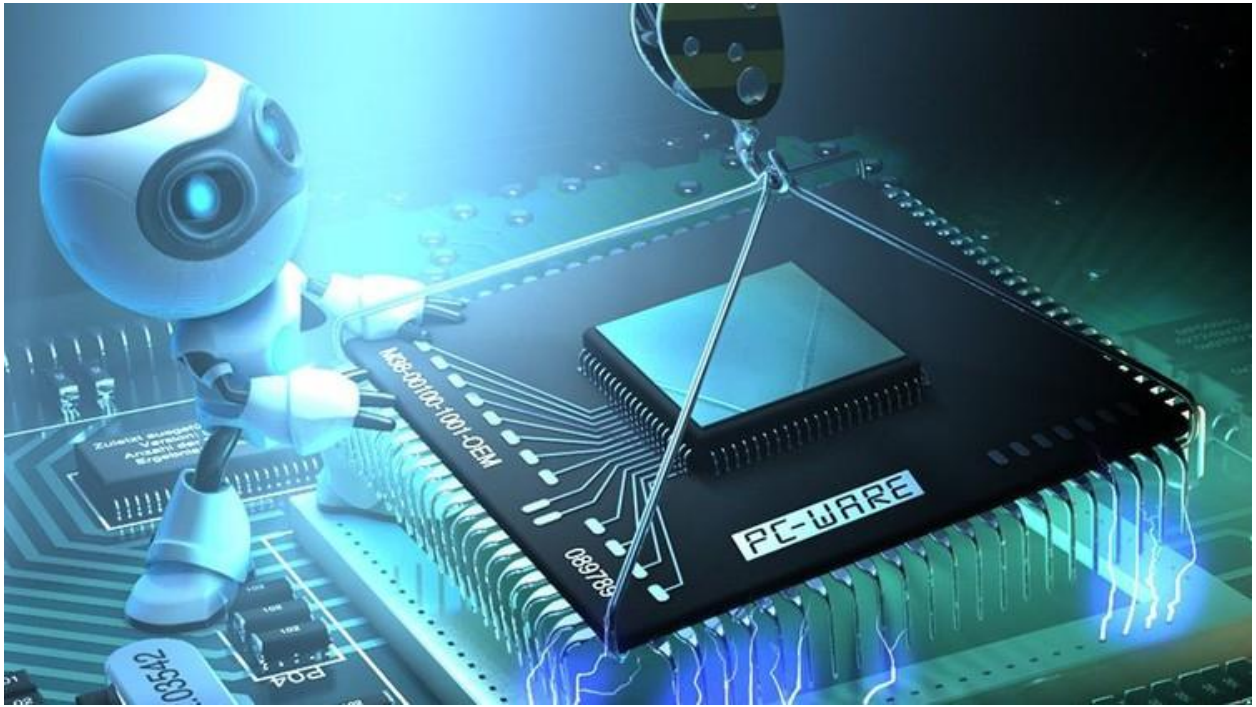
Exam 7 - Problem 1



Exam 7 - Problem 1

Given the following code written in C:

```
int product(int i, int j, int A[100][150], int B[150][200])
{
  int k;
  int sum = 0;
  for (k=0; k<150; k++)
    sum = sum + A[i][k] * B[k][j];
  return sum;
}
```

Exam 7 - Problem 1

Bearing in mind that the function parameters are accessible at the following addresses: **`i`** at **`8(%ebp)`**, **`j`** at **`12(%ebp)`**, that the start address of **`A`** is at **`16(%ebp)`** and the start address of **`B`** at **`20(%ebp)`**, translate the body of the subroutine to x86 assembler. Use registers for the variables **`k`** and **`sum`**.

Exam 7 - Problem 1

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

   movl $0, %eax          ; Initialize sum to 0
   movl $0, %ebx          ; Initialize k to 0

  loop_start:
      cmpl $150, %ebx     ; Compare k with 150
      jge loop_end        ; If k >= 150, exit the loop
```

x86

Part 1/4

Exam 7 - Problem 1

```
; Calculate the memory address for A[i][k]
      movl 8(%ebp), %ecx              ; Load i into %ecx
      movl %ecx, %edx                 ; Copy i into %edx
      imull $600, %edx                ; Multiply i by 600
      addl %edx, %eax                 ; Add 600i to sum
      leal (%eax, %ebx, 4), %ecx      ; Calculate address of A[i][k] in %ecx
      leal 16(%ebp), %edx             ; Load address of @A into %edx
      addl %ecx, %edx                 ; Add the offset to the base address
```

```
sum = sum + A[i][k] * B[k][j];
```

x86

Part 2/4

Exam 7 - Problem 1

```
; Calculate the memory address for B[k][j]
      movl 12(%ebp), %eax              ; Load j into %eax
      imull $800, %ebx                 ; Multiply k by 800
       leal (%eax, %eax, 3), %eax      ; eax = 4j
       addl %eax, %ebx                 ; Add 4j to 800k
       leal 20(%ebp, %ebx, 1), %eax    ; Address of B[k][j] in %eax
```

```
sum = sum + A[i][k] * B[k][j];
```

x86

Part 3/4

Exam 7 - Problem 1

```
; Multiply A[i][k] and B[k][j], and add to sum
      movl (%edx), %ebx          ; Load A[i][k] into %ebx
      imull (%eax), %ebx         ; Multiply A[i][k] with B[k][j]
      addl %ebx, -4(%ebp)        ; Add the result to sum

      incl %ebx                  ; Increment k
      jmp loop_start             ; Continue the loop

   loop_end:
   movl -4(%ebp), %eax           ; Move the final sum into %eax

   popl %ebp
   ret
```

Part 4/4

x86