# 2024-2025 CPT101 quiz

**Question 1**
Correct
Mark 60.00 out of 60.00
Flag question

Suppose you are given an array named myArray containing 5 integers. You task is to develop an assembly program to calculate their sum, and store the result in the ***eax*** register. The instructions you may use to compose this program are provided under the table. Drag-and-drop suitable instructions to fill up line 1 to line 9. Instructions for Lines 3, 4, and 6 are shown already. Complete the rest. (Total 60 marks, i.e. 10 marks for each correct answer.)

| | |
|---|---|
| Line 1 | `lea esi, myArray` ✓ |
| Line 2 | `mov eax, 0` ✓ |
| Line 3 | `mov ecx, 0` |
| Line 4 | `sumLoop:` |
| Line 5 | `add eax, [esi]` ✗ ***#add eax, [esi]#*** |
| Line 6 | `add esi, 4` |
| Line 7 | `inc ecx` ✓ |
| Line 8 | `cmp ecx, 5` ✓ |
| Line 9 | `jl sumLoop` ✓ |

lea esi, myArray | jl sumLoop | add eax, [esi] | dec ecx | cmp ecx, 5 | jnl sumLoop | inc ecx | mov eax, 0 | mov esi, myArray

**Question 2**
Correct
Mark 40.00 out of 40.00
Flag question

What will be the contents of the stack values after the execution of the following code? Assume we have a stack that grows downward. The initial stack pointer points at the memory cell 0x003CFAD6 , and the initial value of each stack space is 00. Dra-and-drop your answer to the end of each stack memory space. (Total 40 marks, i.e. 10 marks for each correct answer.)

```
mov ax, 1
mov bx, 2
mov cx, 3
add bx, ax
sub cx, ax
push bx
push cx
push ax
```

| | |
|---|---|
| 0x003CFACE | 00 ✓ |
| 0x003CFAD0 | 1 ✓ |
| 0x003CFAD2 | 2 ✓ |
| 0x003CFAD4 | 3 ✓ |

0 | 4 | 1 | 00 | 3 | 2

**Question 3**
Partially correct
Mark 60.00 out of 70.00
Flag question

The code segment below is designed to reverse the order of the first 5 characters in a character array named ***myArray***. You may assume the array contains more than 5 characters. Here is how it works:

(1) read a character in the array, starting from the first till the 5th;

(2) push the character into the stack;

(3) repeat steps (1) and (2) five times;

(4) pop a character out from the stack;

(5) place it in the array, starting from the first till the 5th;

(6) repeat steps (4) and (5) five times.

The code segment is incomplete. **Drag-and-drop** the correct arguments and/or instructions to the missing places. (Total 70 marks, i.e. 10 mark for each correct answer.)

| | |
|---|---|
| Line 1 | `start_here: mov ecx, 5` |
| Line 2 | `mov esi,` `0` ✓ |
| Line 3 | `repeat_push:` |
| Line 4 | `mov` ✗ ***#movzx#*** `eax, myArray` `[esi]` ✓ |
| Line 5 | `push eax` |
| Line 6 | `inc` `esi` ✓ |
| Line 7 | `loop` `repeat_push` ✓ |
| Line 8 | `mov esi, 0` |
| Line 9 | `mov ecx, 5` |
| Line 10 | `repeat_pop:pop eax` |
| Line 11 | `mov myArray[esi],` `al` ✓ |
| Line 12 | `inc esi` |
| Line 13 | `loop` `repeat_pop` ✓ |

start_here | repeat_pop | ax | [esi] | 5 | al | eax | 0 | mov | esi | ah | add | dec | movzx | repeat_push | sub

Your answer is partially correct.

**Question 4**

Correct

Mark 30.00 out of 30.00

Flag question

Identify the operand addressing mode used in the following instructions. **Drag-and-drop** your answer to the end of each instruction. (Total 30 marks, i.e. 10 marks for each correct answer.)

```
(1) add ebx, ecx   register ✓
(2) add cx, 2   constant ✓
(3) mov myVar, ebx   memory ✓
```

memory | register | constant | non-addressing | constant+memory | pointer