Research Question: Assembly Language in Modern Computing

Assembly language is a major component of modern computing, especially in areas that require high degrees of efficiency and close hardware connection. Assembly language plays a crucial role in embedded systems, which are essential to industrial automation, automotive, and consumer electronics because it provides accurate control over microcontrollers and microprocessors. The construction of extremely effective and performance-optimized systems is made easier by this direct hardware manipulation, which is essential in situations where computational resources are few.

Additionally, assembly language plays a crucial role in the development of device drivers, which serve as a conduit between an operating system and a computer's hardware. Assembly language is required in this field because it can carry out low-level operations that are inefficiently handled by high-level languages, guaranteeing optimal hardware. both reliability and efficacy.

Another area where assembly language is quite useful is performance optimisation. Assembly language's efficiency can result in significant performance and responsiveness gains in situations where microseconds matter, such high-frequency trading systems or real-time data processing applications. Assembly language lets programmers write code that directly modifies hardware resources, which makes it possible to optimise crucial code segments that high-level languages are unable to accomplish because of their abstraction from the hardware.

Even with its great potential, writing in assembly language is typically more difficult and time-consuming than utilising high-level languages and necessitates a thorough understanding of computer architecture. As a result of this complexity and the development of more potent and effective compilers, assembly language utilisation has decreased. for the creation of generic applications. Assembly language is still a vital tool in the specialised domains of embedded systems, device driver development, and performance-critical applications.

In summary, even though high-level programming languages are becoming more and more common, assembly language's special qualities guarantee that it will always be important in some fields of computer technology. It is a crucial tool for the creation of embedded systems, device drivers, and performance-critical application optimisation due to its direct and effective control over hardware.

Sources:

V. Kanade, "What Is Assembly Language? Working, Features, and Advantages," Spiceworks, 15 June 2023. [Online]. Available: https://www.spiceworks.com/tech/tech-general/articles/what-is-assembly-language.

A. Taylor, "Exploring Assembly Language: An Introduction to Programming," ReversePCB,

November 12, 2022. [Online]. Available: https://reversepcb.com/exploring-assembly-language-an-introduction-to-programming.

AuroraSolutionsAS, "Assembly Language: The Bridge Between Human and Machine," Medium, Mar 15, 2023. [Online]. Available: https://medium.com/@aurorasolutionsas/assembly-language-the-bridge-between-human-and-machine-8624a0d2051c.

```
ction .text
global _start
                                                                                                   1 2 3 4 4 5 6 7 8 9 100 111 12 13 14 15 16 7 18 19 20 1 22 3 24 25 26 29 30 31 32 33 33 34 35 6 37 8 39
 New Project
                                                                                                            start:

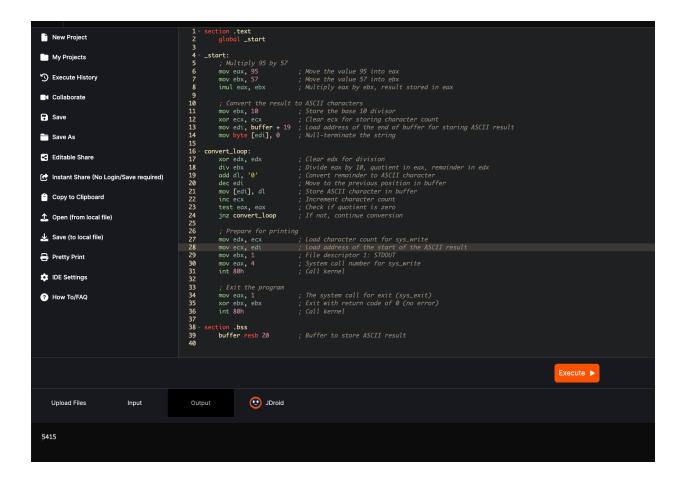
; Subtract 57 from 95

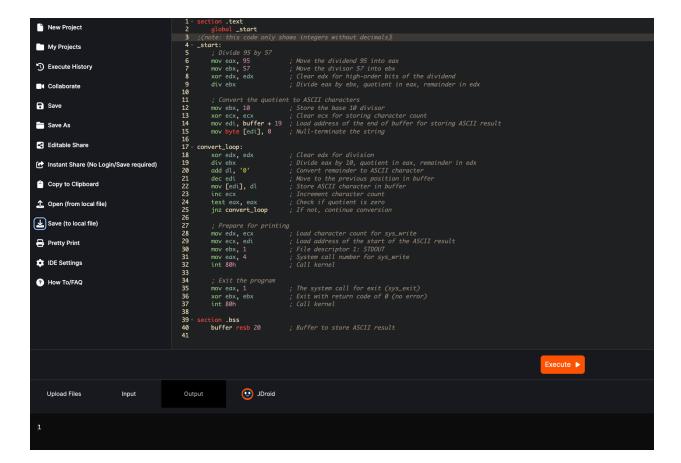
mov eax, 95

; Move the value 95 into eax

sub eax, 57

; Subtract 57 from eax
My Projects
S Execute History
                                                                                                                       ; Convert the result to ASCII characters
mov ebx, 10 ; Store the base 10 divisor
xor ecx, ecx ; Clear ecx for storing character count
mov edt, buffer + 19 ; Load address of the end of buffer for storing ASCII result
mov byte [edi], 0 ; Null-terminate the string
■ Collaborate
 ■ Save
ave As
                                                                                                             convert_loop:
                                                                                                                                                                     ; Clear edx for division
; Divide eax by 10
; Convert remainder to ASCII character
; Move to the previous position in buffer
; Store ASCII character in buffer
; Increment character count
; Check if quotient is zero
; If not, continue conversion
                                                                                                                       ivert_loop:
    xor edx, edx
    div ebx
    add dl, '0'
    dec edi
    mov [edi], dl
    inc ecx
    test eax, eax
    jnz convert_loop
 ditable Share
Instant Share (No Login/Save required)
Copy to Clipboard
① Open (from local file)
                                                                                                                       ; Prepare for printing
mov edx, ecx ; Load character count for sys_write
mov ecx, edi ; Load address of the start of the ASCII result
mov ebx, 1 ; File descriptor 1: STDOUT
mov eax, 4 ; System call number for sys_write
int 80h ; Call kernel
 Save (to local file)
 Pretty Print
 IDE Settings
                                                                                                                       ; Exit the program mov eax, 1 xor ebx, ebx int 80h
 ? How To/FAQ
                                                                                                                       ction .bss
buffer resb 20
                                                                                                                                              JDroid
       Upload Files
                                                          Input
                                                                                                      Output
```





```
ction .data
msg db "Original String: "
len equ $-msg ; Calculates msg string's length
New Project
                                                                                                                                                                                                                    msg2 @ 0ah ,"Reversed string: " ; Adds newline before "Reversed String: "
lenMsg2 equ $ - msg2 ; Computes length of msg2 string
                                                                                                                                                                                                                                                                      msg3 db 0ah, "Uppercase String: "
lenHsg3 equ $-msg3 ; Deterwines length of msg3 string
                                                                                                                                                                                                                                                                      firstString db "team"
lenFirst equ $-firstString ; Measures length of firstString
                                                                                                                                                                                                                                                                      SecondString db "mate"
lenSecond equ $-SecondString ; Measures length of SecondString
Copy to Clipboard
↑ Open (from local file)
Save (to local file)
                                                                                                                                                                                                                                                            Int 8th , Resets obx to 0
; Capies characters from source to destination until count reaches zero occup.loop:
nov al. [est] ; Retrieves character from source
nov [adl], al ; Stores character in destination
to est ; Admices source profiter
des ex ; Reduces loop counts pointer
in copy.loop ; Continues if count
                                                                                                                                                                                                                                                                  call writerising

; Prepares to capp firstString into result
and est, firstString
and est, leafirst
; Loop count set to firstString's length
call copy_loop

; Appends SecondString to result after firstString
and est, SecondString
and est, SecondString
and est, SecondString
and est, est
and e
                                                                                                                                                                                                                                                                  call writeSecondMsg
```

```
; Sets up for reversing the concatenated string
mov est, result
mov edt, result
det edt ; Adjusts edt to point at string's end
mov ecx, lenFirst + lenSecond ; Count set to concatenated string length
                                                                            109 ; Sets up
110 mov est, p
1111 mov est, p
1112 dec edt
113 mov ex, p
115 call print
116 call write
117 call write
118 ; Converts
120 mov est, p
121 convert, p
1221 convert, p
1223 cmp ol
123 cmp ol
124 je con
125 je not
126 ; Chec
127 cmp ol
128 jb not
129 cmp
130 ja not
131 ja not
131 ja not
132 ; Converts
133 sub ol
134 sub ol
135 not_lowercase:
136 mov [e
137 convert_done:
142 convert_done:
142 convert_done:
143 call print
144 convert_done:
145 coll exit_l
146
                                                                                                call reverse_string_loop
call print_result
                                                                                                call writeThirdMsg
                                                                                               ; Converts concatenated and reversed string to uppercase
mov esi, result
convert.to.uppercase:
mov al, [esi] ; Loads current character
cmp al, 9 ; Checks for end of string
je convert_done ; Ends loop if end of string found
                                                                                                        cmp al, 'a'
jb not_lowercase
                                                                                                       cmp al, 'z'
ja not_lowercase
                                                                                                       ; Converts lomercase to uppercase by adjusting ASCII value sub al, 32
                                                                                       not_lowercase:
    mov [esi], al    ; Saves converted character
                                                                                                       call print_result
                                                                                                call exit_program
                                                                                                             JDroid
                                                                              Output
     Upload Files
                                            Input
Original String: teammate
Reversed string: etammaet
Uppercase String: ETAMMAET
                                                                                New Project
My Projects
Execute History
■ Collaborate
■ Save
Save As
                                                                                                  ; Prepare for printing the concatenated string
mov eax, 4
; System call number for sys_write
mov ebx, 1
; File descriptor 1 (stdout)
mov ecx, buffer
mov edx, 8
; Pointer to the concatenated string in buffer
mov edx, 8
; Length of the concatenated string (6 bytes for "Team" + 8 bytes for "Mate")
int 0x80
; Invoke syscall
Editable Share
[ Instant Share (No Login/Save required)
Copy to Clipboard
① Open (from local file)

▲ Save (to local file)

Pretty Print
IDE Settings
? How To/FAQ
     Upload Files
                                                                                                                    JDroid
                                                                                                                                                                                                                                                                                                                  Generated Files
```

