Farrel Ganendra | 5024231036

Pertama tama, saya install DOSBox di https://www.dosbox.com/download.php?main=1

Kemudian, saya menambahkan DOSBox ke variable environtment windows supaya bisa dijalankan langsung dimanamun dari terminal. Ini berguna supaya nanti mempermudah kita menggunakan file konfigurasi dosbox buatan kita sendiri. Normalnya, kita perlu melakukan mount ke folder yang berisi file assembly kita (dan beberapa software penting seperti tasm, tlink, dan ms-debug). Untuk membuat DOSBox melakukan hal itu secara automatis, kita copy dan paste file dosbox.conf ke folder workspace kita dan tambahkan beberapa line berikut diakhir file.

```
mount C C:\Users\farre\projects\penjumlahan_matriks

C:

set PATH=%PATH%;C:\compiler;C:\compiler\DEBUGX
```

Kemudian, kita bisa menjalankan DOSBox dengan perintah

```
Dosbox -noconsole -conf .\dosbox.conf
```

Kemudian disini kita bisa melakukan praktikum, kita bisa membuat code assembly dengan program EDIT atau menggunakan vscode. Saya memilih menggunakan VSCode. Pertama saya membuat program assembly JUMLAH.asm. Di program ini, isi matriks dan juga n (panjang matriks) nya ditetapkan secara hardcode di segment data di line 2 - 7.

JUMLAH.asm

You, 3 seconds ago | 1 author (You)

```
1  .model small
2  .data
3      n db 20                                                    ; Number of elements to add
4      A db 0, 1, 1, 2, 3, 3, 4, 4, 5, -1, 1, 2, 2, 2, 3, 3, 4, 4, 0, 1 ; Matrix A      You, 1 second ago •
5      B db 1, 1, 2, 2, 2, 3, 3, 4, 4, 0, 1, 1, 2, 3, 3, 4, 4, 5, 1, 1  ; Matrix B
6      C db 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  ; Matrix C to store the result
7      newline db 13, 10, '$'                                     ; Newline characters for printing
```

Apabila ingin mengubah matriks, dapat dilakukan dengan mengubah file JUMLAH.asm nya. Kemudian untuk melakukan kompilasi, kita bisa menggunakan turbo assembler dengan menuliskan perintah berikut di jendela DOSBox:

```
tasm JUMLAH.asm

tlink JUMLAH.OBJ
```

Kemudian kita bisa menjalankan program dengan perintah “JUMLAH.exe”

Berikut merupakan 20 karakter isi dari matriks C. Terdapat karakter aneh ‘/’. Hal ini karena pada elemen tersebut, penjumlahan yang dilakukan adalah A[9] + B[9] = (-1) + 0 = -1. Hasil penjumlahan -1 ini oleh program di konversi menjadi karakter ASCII yang berada tepat sebelum karakter 0 yaitu ‘/’. Dengan perintah DEBUG, kita bisa mengamati step by step bagaimana program berjalan.

```
C:\>debug JUMLAH.exe
-t
AX=06D4 BX=0000 CX=0080 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06C0 ES=06C0 SS=06CF CS=06D0 IP=0003 NV UP EI PL NZ NA PO NC
06D0:0003 8ED8              MOV     DS,AX
-t
AX=06D4 BX=0000 CX=0080 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0005 NV UP EI PL NZ NA PO NC
06D0:0005 8A0E0000          MOV     CL,[0000]                       DS:0000=14
-t
AX=06D4 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0009 NV UP EI PL NZ NA PO NC
06D0:0009 BE0000            MOV     SI,0000
```

Disini kita lihat bahwa Langkah pertama yang program lakukan adalah memasukkan data data n, matriks A, B, dan C ke register DS. Kemudian program mengisi nilai 0 ke register CL atau bagian 8 bit bawah dari register counter CX. Hal ini dilakukan karena program ingin melakukan looping, Untuk operasi yang memerlukan index, program saya menggunakan register SI. Sehingga tentunya Langkah selanjutnya adalah menyetel isi register SI menjadi 0. Selanjutnya, program akan melakukan looping sebagai berikut.

```
-p
AX=06D4 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=000C NV UP EI PL NZ NA PO NC
06D0:000C 8A840100          MOV     AL,[SI+0001]                    DS:0001=00
-p
AX=0600 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0010 NV UP EI PL NZ NA PO NC
06D0:0010 02841500          ADD     AL,[SI+0015]                    DS:0015=01
-p
AX=0601 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0014 NV UP EI PL NZ NA PO NC
06D0:0014 88842900          MOV     [SI+0029],AL                    DS:0029=00
-p
AX=0601 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0018 NV UP EI PL NZ NA PO NC
06D0:0018 46                INC     SI
-p
AX=0601 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0001 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0019 NV UP EI PL NZ NA PO NC
06D0:0019 E2F1              LOOPW   000C
```

Disini kita lihat dalam loop, pertama program menggunakan register AL atau bagian 8 bit bawah dari register akumulator AX sebagai tempat untuk melakukan penjumlahan. Pertama tama, nilai matriks A yang berada di address 0001 + SI dengan SI = 0 (SI sebagai offset atau index) dimasukkan ke AL, kemudian, program menambahkan isi AL dengan nilai matriks B yang berada di address 0015 + SI dengan SI = 0. Setelah itu, Kita memasukkan hasil penjumlahan yang masih berada di register akumulator kedalam Matriks C yang berada pada 0029 + SI dengan SI juga 0. Kemudian, Program melakukan Increment ke register SI sehingga SI menjadi 1 dan lalu melakukan LOOP Kembali ke instruksi di address 000C yang merupakan bagian awal dari loop (MOV AL, [A + SI]) dan program pun siap melakukan penjumlahan untuk elemen matriks selanjutnya. Setiap perulangan, register counter CX akan di decrement secara automatis oleh mikroprosesor (emulated) sehingga ketika register CX bernilai 0, maka perulangan akan berhenti dan kita akan menuju ke Langkah selanjutnya setelah looping. Setelah looping selesai, program Bersiap siap untuk melakukan print isi dari matriks C.

```
-p
AX=0600 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0014 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=001B NV UP EI PL NZ NA PE CY
06D0:001B B100              MOV     CL,00
-t
AX=0600 BX=0000 CX=0000 DX=0000 SP=0000 BP=0000 SI=0014 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=001D NV UP EI PL NZ NA PE CY
06D0:001D 8A0E0000          MOV     CL,[0000]                       DS:0000=14
-t
AX=0600 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0014 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0021 NV UP EI PL NZ NA PE CY
06D0:0021 BE0000            MOV     SI,0000
-p
AX=0600 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0024 NV UP EI PL NZ NA PE CY
06D0:0024 8A842900          MOV     AL,[SI+0029]                    DS:0029=01
-p
AX=0601 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0028 NV UP EI PL NZ NA PE CY
06D0:0028 0430              ADD     AL,30
-t
AX=0631 BX=0000 CX=0014 DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=002A NV UP EI PL NZ NA PO NC
06D0:002A 8AD0              MOV     DL,AL
-S_
```

```
-t
AX=0631 BX=0000 CX=0014 DX=0031 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=002C NV UP EI PL NZ NA PO NC
06D0:002C B402              MOV     AH,02
-t
AX=0231 BX=0000 CX=0014 DX=0031 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=002E NV UP EI PL NZ NA PO NC
06D0:002E CD21              INT     21
-t
AX=0231 BX=0000 CX=0014 DX=0031 SP=0000 BP=0000 SI=0000 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0030 NV UP EI PL NZ NA PO NC
06D0:0030 46                INC     SI
-t
AX=0231 BX=0000 CX=0014 DX=0031 SP=0000 BP=0000 SI=0001 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0031 NV UP EI PL NZ NA PO NC
06D0:0031 E2F1              LOOPW   0024
-pS
```

Disini dapat kita lihat pertama program mengosongkan register counter dan mengisinya ulang dengan data n yang berada pada address relative 0000 dan juga mereset register SI yang digunakan sebagai index Kembali menjadi 0. Lalu program masuk kedalam loop. Didalam loop, program menggunakan register AX untuk mengonversi bilangan ke karakter ASCII (konversi ini hanyalah operasi pejumlahan biasa), setiap elemen di karakter C dimasukkan ke AL satu persatu dan ditambahkan dengan nilai 30 untuk mengubahnya ke nilai ASCII. Kemudian, hasilnya dimasukkan ke register DL yang merupakan register DATA. Selagi angka yang ingin kita print berada di register ini, kita dapat melakukan interrupt 02h untuk melakukan printing. Kemudian program melakukan increment pada index dan kemudian melanjutkan ke siklus loop selanjutnya hingga selesai. Setelah loop selesai, dapat kita lihat

```
3456789/2345678910AX=0230 BX=0000 CX=0000 DX=0030 SP=0000 BP=0000 SI=0014 DI=000
0
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0033 NV UP EI PL NZ NA PE NC
06D0:0033 BA3D00            MOV     DX,003D
-t
AX=0230 BX=0000 CX=0000 DX=003D SP=0000 BP=0000 SI=0014 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0036 NV UP EI PL NZ NA PE NC
06D0:0036 B409              MOV     AH,09
-t
AX=0930 BX=0000 CX=0000 DX=003D SP=0000 BP=0000 SI=0014 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=0038 NV UP EI PL NZ NA PE NC
06D0:0038 CD21              INT     21
-t

AX=0930 BX=0000 CX=0000 DX=003D SP=0000 BP=0000 SI=0014 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=003A NV UP EI PL NZ NA PE NC
06D0:003A B8004C            MOV     AX,4C00
-t
AX=4C00 BX=0000 CX=0000 DX=003D SP=0000 BP=0000 SI=0014 DI=0000
DS=06D4 ES=06C0 SS=06CF CS=06D0 IP=003D NV UP EI PL NZ NA PE NC
06D0:003D CD21              INT     21
-t

Program terminated (0000)
-S
```

Disini program melakukan print character newline dan kemudian mengubah akumulator ke 4C00 dan melakukan interrupt untuk exit dari program. Program keluar dengan exit code 0 yang berarti tidak terjadi error.

Di program JUMLAH-CALL.exe, logika yang sama digunakan untuk melakukan penjumalahan namun di implementasi dengan prosedur sendiri yang kemudian di CALL di procedure main.

```
56  ; Subroutine to add matrices A and B
57  add_matrices proc
58      mov cx, 0               ; Clear high byte of CX
59      mov cl, n               ; Set loop counter to the number of elements
60      mov si, 0               ; Index for accessing elements
61
62  jumlah:
63      mov al, [A + si]        ; Load A[si] into AL
64      add al, [B + si]        ; Add B[si]
65      mov [C + si], al        ; Store result in C[si]
66      inc si                  ; Increment index
67      loop jumlah             ; Loop until CX is 0
68      ret
69  add_matrices endp
```

Terdapat juga 2 subrutin lain yaitu subrutin untuk mengisi Matriks A dan B. di prosedur main. Yang dilakukan adalah meminta input untuk nilai n, kemudian memanggil subrutin isi A, isi B, dan subrutin penjumlahan matriks. Kemudian program melakukan print ke terminal dan exit. Berikut implementasi salah satu subrutin pengisian matriks

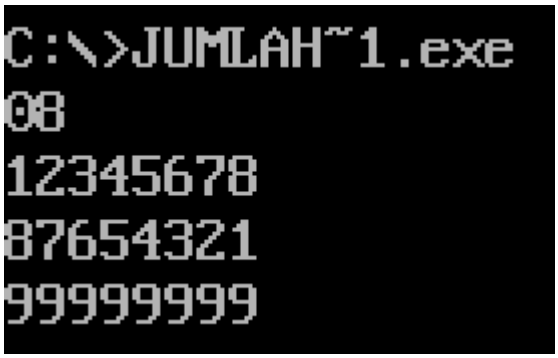
```
; Subroutine to input matrix A
input_A proc
    mov cx, 0               ; Clear high byte of CX
    mov cl, n               ; Set loop counter to the number of elements
    mov si, 0               ; Index for accessing elements

input_A_loop:
    ; Prompt for input A[si]
    mov ah, 01h             ; DOS interrupt to read a character
    int 21h                 ; Read character into AL
    sub al, 30h             ; Convert ASCII to numeric value (assuming input is 0-9)
    mov [A + si], al        ; Store input in A[si]
    inc si                  ; Increment index
    loop input_A_loop       ; Loop until CX is 0

    ; Print newline
    lea dx, newline         ; Load address of newline characters
    mov ah, 09h             ; DOS interrupt to print string
    int 21h                 ; Print newline

    ret
input_A endp
```

Untuk menjalankan program, perlu kita ingat bahwa DOSBox menyingkat nama file yang panjangnya lebih dari 5 karakter menjadi 6 karakter dengan format nama~1. Misal, nama file JUMLAH-CALL.asm menjadi JUMLAH~1. Jadi pastikan untuk menggunakan nama file yang pendek atau sesuai dengan format tersebut saat menjalankan program. Atau cek nama file dengan perintah dir di jendela DOSBox. Sehingga untuk melakukan kompilasi, kita bisa menggunakan perintah berikut

tasm JUMLAH~1.asm

tlink JUMLAH~1.OBJ

Lalu ketika kita menjalankan program, kita akan diminta 3 input. Input pertama adalah 2 digit yang merepresentasikan n, kemudian input kedua dan ketiga adalah isi dari matriks A dan B. Format input tidak boleh ada spasi, untuk bilangan n 1 digit, wajib berikan angka 0 didepannya. dan jangan klik enter setelah selesai memasukkan salah satu input (misal setelah selesai memasukkan input n), program akan berpindah ke new line dengan automatis. Hal ini karena Ketika kita mengklik enter di keyboard kita untuk berpindah ke line bawah di terminal, klik tersebut terhitung sebagai input yang masuk ke register dx juga dan dibaca oleh program yang mengira itu adalah value ASCII yang perlu dijadikan angka, hal ini bisa mengakibatkan prompt input yang terlihat seperti lebih sedikit dari seharusnya dan output yang berisikan karakter karakter aneh. Berikut hasil menjalankan programnya



Apabila kita melakukan DEBUG terhadap program ini, kita dapat melihat beberapa perbedaan dari program pertama.

```
-t
AX=06DA BX=0000 CX=055E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06C0 ES=06C0 SS=06CF CS=06D0 IP=005B NV UP EI PL NZ NA PO NC
06D0:005B 8ED8              MOV     DS,AX
-t
AX=06DA BX=0000 CX=055E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=005D NV UP EI PL NZ NA PO NC
06D0:005D B401              MOV     AH,01
-t
AX=01DA BX=0000 CX=055E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=005F NV UP EI PL NZ NA PO NC
06D0:005F CD21              INT     21
-t
0AX=0130 BX=0000 CX=055E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0061 NV UP EI PL NZ NA PO NC
06D0:0061 2C30              SUB     AL,30
-t
AX=0100 BX=0000 CX=055E DX=0000 SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0063 NV UP EI PL ZR NA PE NC
06D0:0063 B20A              MOV     DL,0A
-t
AX=0100 BX=0000 CX=055E DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0065 NV UP EI PL ZR NA PE NC
06D0:0065 F6E2              MUL     DL
-S_
```

```
-t
AX=0000 BX=0000 CX=055E DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0067 NV UP EI PL ZR NA PE NC
06D0:0067 A20A00            MOV     [000A],AL                       DS:000A=14
-t
AX=0000 BX=0000 CX=055E DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=006A NV UP EI PL ZR NA PE NC
06D0:006A B401              MOV     AH,01
-t
AX=0100 BX=0000 CX=055E DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=006C NV UP EI PL ZR NA PE NC
06D0:006C CD21              INT     21
-t
3AX=0133 BX=0000 CX=055E DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=006E NV UP EI PL ZR NA PE NC
06D0:006E 2C30              SUB     AL,30
-t
AX=0103 BX=0000 CX=055E DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0070 NV UP EI PL NZ NA PE NC
06D0:0070 00060A00          ADD     [000A],AL                       DS:000A=00
-t
AX=0103 BX=0000 CX=055E DX=000A SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0074 NV UP EI PL NZ NA PE NC
06D0:0074 BABB04            MOV     DX,04BB
-S_
```

```
-t
AX=0103 BX=0000 CX=055E DX=04BB SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0077 NV UP EI PL NZ NA PE NC
06D0:0077 B409              MOV     AH,09
-t
AX=0903 BX=0000 CX=055E DX=04BB SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0079 NV UP EI PL NZ NA PE NC
06D0:0079 CD21              INT     21
-t

AX=0903 BX=0000 CX=055E DX=04BB SP=0000 BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=007B NV UP EI PL NZ NA PE NC
06D0:007B E882FF            CALL    0000
-S_
```

Disini dapat kita lihat seperti biasa program mula mula memasukkan data ke register DS. Kemudian kita melakukan interrupt untuk user input dengan cara memasukkan AX high (AH) ke nilai 1 dan mentrigger interrupt. Dapat dilihat saya memasukkan nilai 0 (dibagian yang diberikan kotak merah). Setelah interrupt dijalankan, maka register AX low (AL) akan berisi nilai ASCII user input yang akan kita jadikan sebagai nilai n. nilai n ini kita konversikan dari nilai ASCII ke nilai numerik biasa dengan cara menguranginya dengan 30 (SUB DL, 30). Karena nilai ini adalah digit pertama dari input n yang terdiri dari 2 digit, maka kita perlu mengalikannya dengan 10. Hal ini bisa kita lakukan dengan cara memasukkan nilai 10 ke register DX (DX low atau DL karena angka 8 bit) dan lalu menggunakan intruksi MUL (MUL DL). Instruksi ini akan mengalikan operand (DL) ke isi register AX yang sekarang sedang diisi oleh angka digit pertama n kita. Kemudian kita menyimpan nilai di AX ke n karena kita memerlukan register AX untuk input digit kedua. Dengan cara yang sama, saya memasukkan nilai 3 (di kotak merah) dan lalu program melakukan konversi dari ASCII dan menambahkan nilai tersebut ke n yang berada di address relative 000A. Kemudian program pun melakukan print newline. Dan melakukan panggilan ke subroutine input A yang berada pada address 0000. Berikut program menjalankan subrutin input A

```
-t
AX=0903 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0003 NV UP EI PL NZ NA PE NC
06D0:0003 8A0E0A00          MOV     CL,[000A]                       DS:000A=03
-t
AX=0903 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0007 NV UP EI PL NZ NA PE NC
06D0:0007 BE0000            MOV     SI,0000
-t
AX=0903 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=000A NV UP EI PL NZ NA PE NC
06D0:000A B401              MOV     AH,01
-t
AX=0103 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=000C NV UP EI PL NZ NA PE NC
06D0:000C CD21              INT     21
-t
AX=0174 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=000E NV UP EI PL NZ NA PE NC
06D0:000E 2C30              SUB     AL,30
-
AX=0144 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0010 NV UP EI PL NZ NA PE NC
06D0:0010 88840B00          MOV     [SI+000B],AL                    DS:000B=00
-S
```

```
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0010 NV UP EI PL NZ NA PE NC
06D0:0010 88840B00          MOV     [SI+000B],AL                    DS:000B=00
-t
AX=0144 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0014 NV UP EI PL NZ NA PE NC
06D0:0014 46                INC     SI
-t
AX=0144 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0001 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0015 NV UP EI PL NZ NA PO NC
06D0:0015 E2F3              LOOPW   000A
-p
11AX=0101 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0017 NV UP EI PL NZ NA PE NC
06D0:0017 BABB04            MOV     DX,04BB
-      ╓ ╓ ╓ ╓ z ╓ ╓ ╓ ↓ ¶ ↕p
      ^ Error
-t
AX=0101 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=001A NV UP EI PL NZ NA PE NC
06D0:001A B409              MOV     AH,09
-t
AX=0901 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=001C NV UP EI PL NZ NA PE NC
06D0:001C CD21              INT     21
-S
```

```
AX=0901 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=001E NV UP EI PL NZ NA PE NC
06D0:001E C3                RET
-S_
```

Disini dapat kita lihat pertama program mempersiapkan looping dengan mereset register CX ke n dan SI ke 0. Lalu, kita mulai melakukan interrupt untuk input, mengonversi nya dari nilai ASCII, dan lalu menyimpannya ke matriks A yang berada di address relative 000B + si. Lalu program melakukan increment ke register si dan melakukan looping dengan cara Kembali ke intruksi 000A atau intruksi awal loop (MOV AH, 01) yang mengatur interrupt untuk input character. Diluar loop, program mengeprint new line dan melakukan RET yang akan Kembali ke dimana subrutin ini dipanggil.

```
AX=0901 BX=0000 CX=0000 DX=04BB SP=0000 BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=007E NV UP EI PL NZ NA PE NC
06D0:007E E89EFF            CALL    001F
-p
111
AX=0901 BX=0000 CX=0000 DX=04BB SP=0000 BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0081 NV UP EI PL NZ NA PE NC
06D0:0081 E8BAFF            CALL    003E
-S
```

Setelah return, program akan langsung memanggil subrutin input B yang berada pada address 001F. Karena isi program nya sama Degnan subrutin A, saya langsung skip dengan perintah proceed ‘p’ dan menginput nilai nya [1, 1, 1]. Setelah itu program langsung memanggil CALL ke subrutin penjumlahan dua matriks yang berada di 003E. Seperti biasa, program akan Bersiap untuk looping

```
-t
AX=0901 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=003E NV UP EI PL NZ NA PE NC
06D0:003E B90000            MOV     CX,0000
-t
AX=0901 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0041 NV UP EI PL NZ NA PE NC
06D0:0041 8A0E0A00          MOV     CL,[000A]                       DS:000A=03
-t
AX=0901 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0045 NV UP EI PL NZ NA PE NC
06D0:0045 BE0000            MOV     SI,0000
```

Kemudian, melakukan loop untuk menjumlahkan tiap elemen A dan B dan menyimpannya di matriks C

```
-t
AX=0901 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0048 NV UP EI PL NZ NA PE NC
06D0:0048 8A840B00          MOV     AL,[SI+000B]                    DS:000B=44
-t
AX=0944 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=004C NV UP EI PL NZ NA PE NC
06D0:004C 02849B01          ADD     AL,[SI+019B]                    DS:019B=01
-t
AX=0945 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0050 NV UP EI PL NZ NA PO NC
06D0:0050 88842B03          MOV     [SI+032B],AL                    DS:032B=00
-t
AX=0945 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0000 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0054 NV UP EI PL NZ NA PO NC
06D0:0054 46                INC     SI
-t
AX=0945 BX=0000 CX=0003 DX=04BB SP=FFFE BP=0000 SI=0001 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0055 NV UP EI PL NZ NA PO NC
06D0:0055 E2F1              LOOPW   0048
-p
AX=0902 BX=0000 CX=0000 DX=04BB SP=FFFE BP=0000 SI=0003 DI=0000
DS=06DA ES=06C0 SS=06CF CS=06D0 IP=0057 NV UP EI PL NZ NA PE NC
06D0:0057 C3                RET
-S
```

Disini penjumlahan dilakukan dengan cara yang sama seperti pada program JUMLAH.asm dan lalu melakukan return. Kemudian program akan mulai melakukan print ke hasil yang memiliki logika yang juga sama dengan di program JUMLAH.asm dan lalu exit.

Disini output angka pertama di matriks C adalah u sedangkan angka kedua dan ketigaa adalah 2. Hal ini terjadi karena saya tidak sengaja memasukkan nilai ‘t’ untuk elemen pertama matriks A.