Microprocessors

Final project (software)

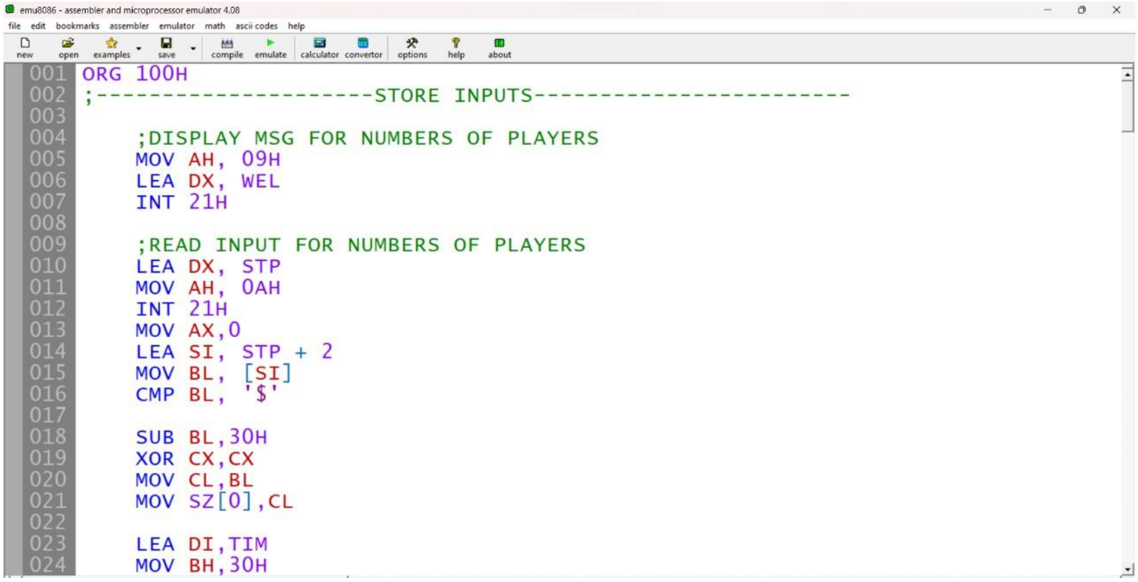
Marathon results

Name	ID
Mohamed Hosny Qasem	21011117
Ziad Ismail Hassan	21010559
Adham Abu Bakr Morsy Mohamed	21010219
Nardeen Refaat Fuad	20012076
Asmaa Hassan Mokhtar Aboushady	19015430

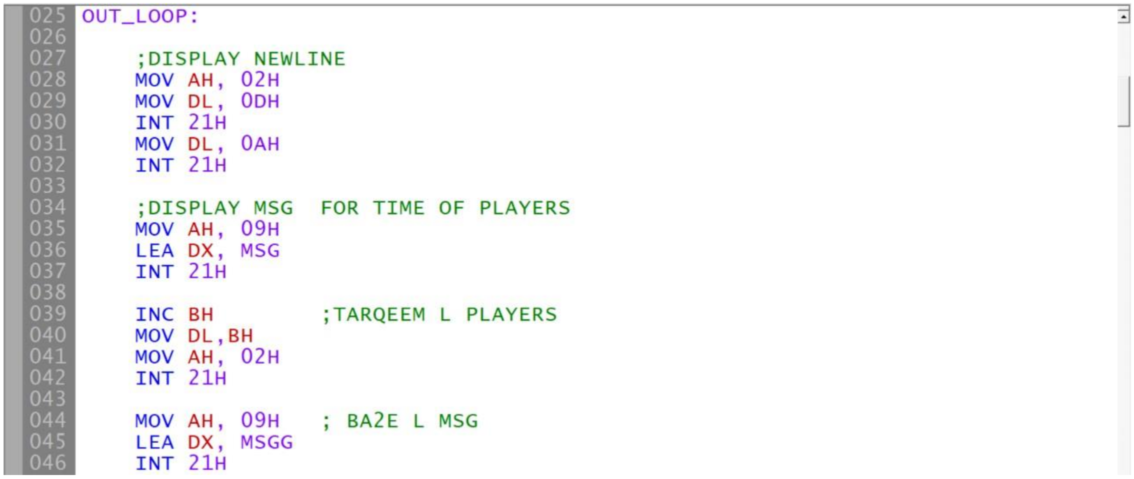
Marathon Results

There are N players participating in a marathon. Their numbers and time in which they completed the marathon are stored in the memory. It is required to rearrange them in ascending order to find the winner. The inputs are the number of players and two tables. The first table contains the player number, and the second one contains their recorded time. The outputs are two tables. The first one contains the player number arranged according to their times and the second table shows these times.

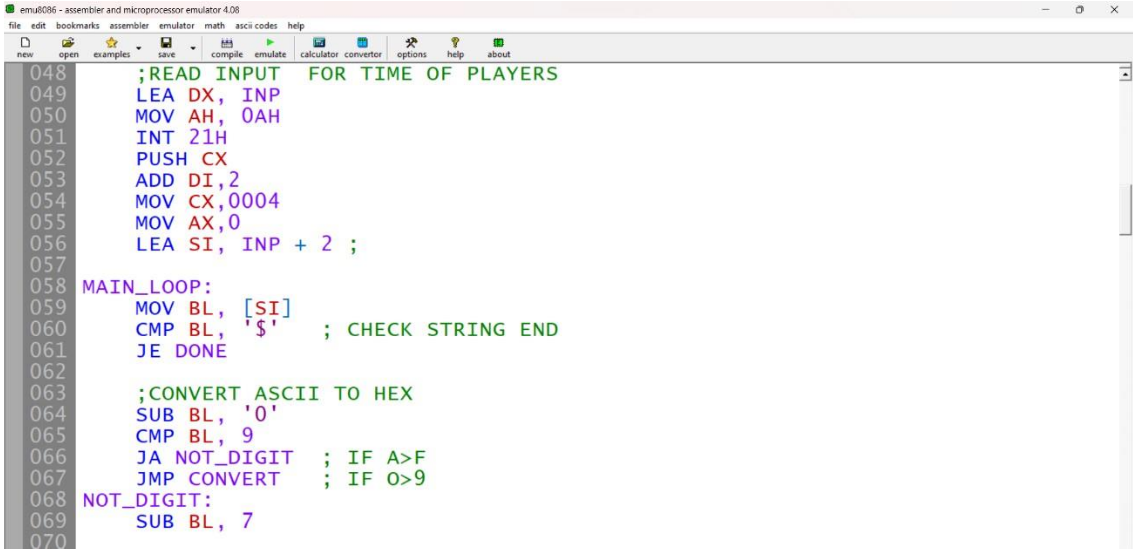
The code:



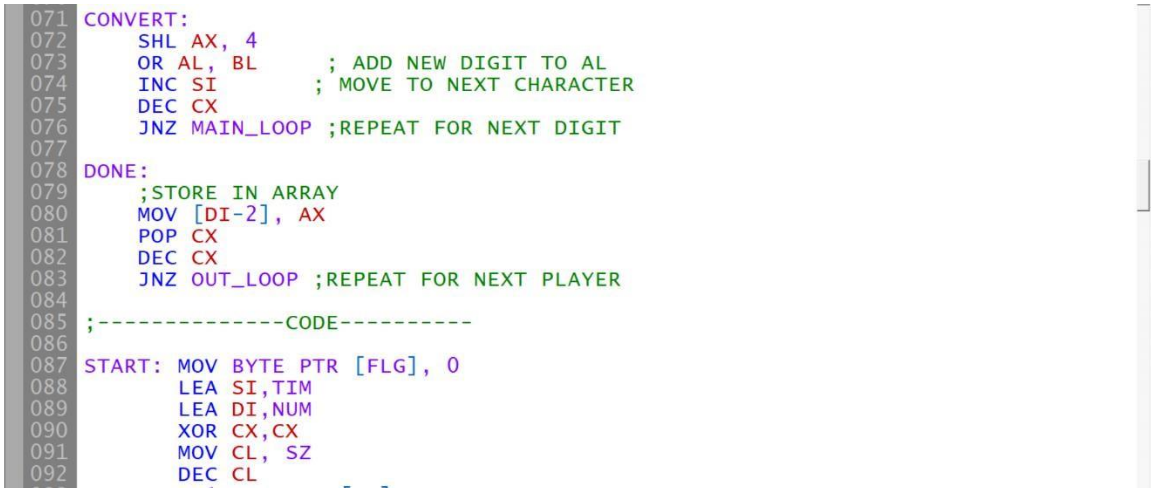
```
ORG 100H
;---------------------STORE INPUTS------------------------

    ;DISPLAY MSG FOR NUMBERS OF PLAYERS
    MOV AH, 09H
    LEA DX, WEL
    INT 21H

    ;READ INPUT FOR NUMBERS OF PLAYERS
    LEA DX, STP
    MOV AH, 0AH
    INT 21H
    MOV AX,0
    LEA SI, STP + 2
    MOV BL, [SI]
    CMP BL, '$'

    SUB BL,30H
    XOR CX,CX
    MOV CL,BL
    MOV SZ[0],CL

    LEA DI,TIM
    MOV BH,30H
```



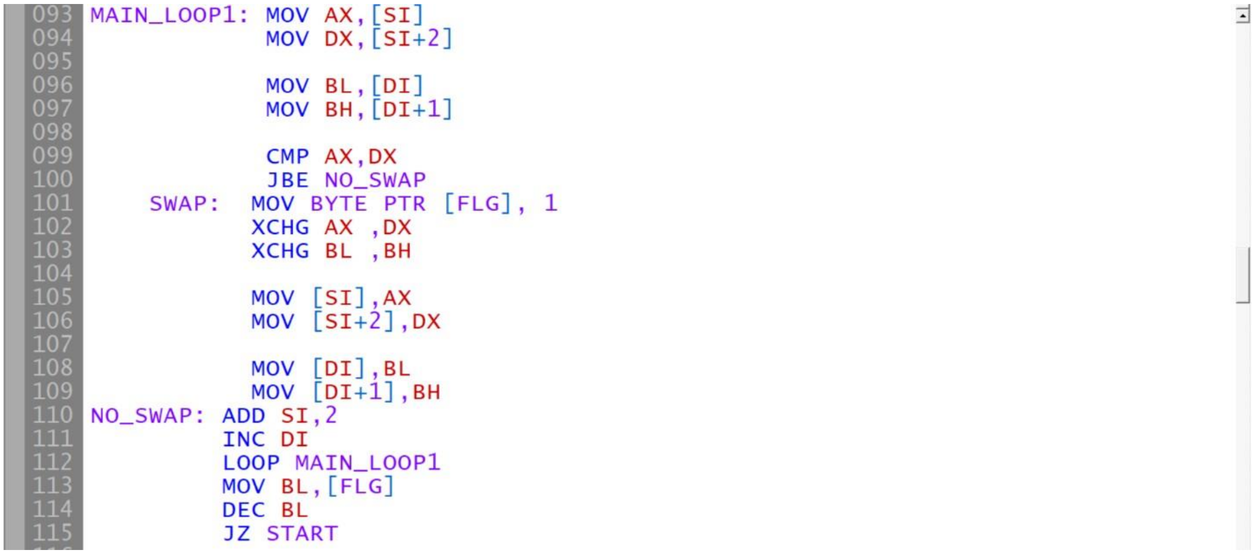
```
OUT_LOOP:

    ;DISPLAY NEWLINE
    MOV AH, 02H
    MOV DL, 0DH
    INT 21H
    MOV DL, 0AH
    INT 21H

    ;DISPLAY MSG  FOR TIME OF PLAYERS
    MOV AH, 09H
    LEA DX, MSG
    INT 21H

    INC BH          ;TARQEEM L PLAYERS
    MOV DL,BH
    MOV AH, 02H
    INT 21H

    MOV AH, 09H   ; BA2E L MSG
    LEA DX, MSGG
    INT 21H
```


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new
open
examples
save
compile
emulate
calculator
convertor
options
help
about

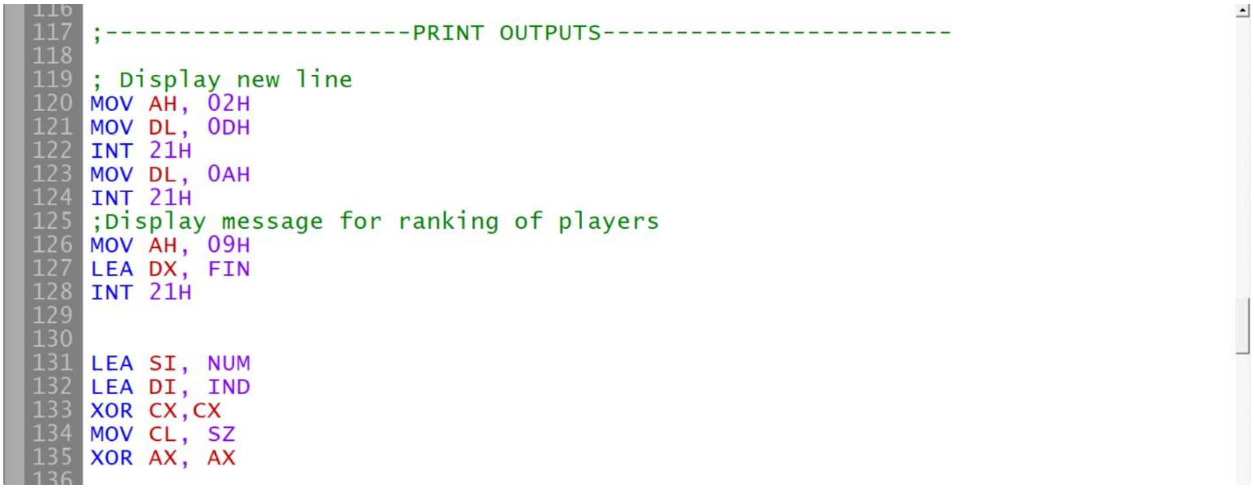
```
    ;READ INPUT  FOR TIME OF PLAYERS
    LEA DX, INP
    MOV AH, 0AH
    INT 21H
    PUSH CX
    ADD DI,2
    MOV CX,0004
    MOV AX,0
    LEA SI, INP + 2 ;

MAIN_LOOP:
    MOV BL, [SI]
    CMP BL, '$'   ; CHECK STRING END
    JE DONE

    ;CONVERT ASCII TO HEX
    SUB BL, '0'
    CMP BL, 9
    JA NOT_DIGIT  ; IF A>F
    JMP CONVERT   ; IF 0>9
NOT_DIGIT:
    SUB BL, 7

```



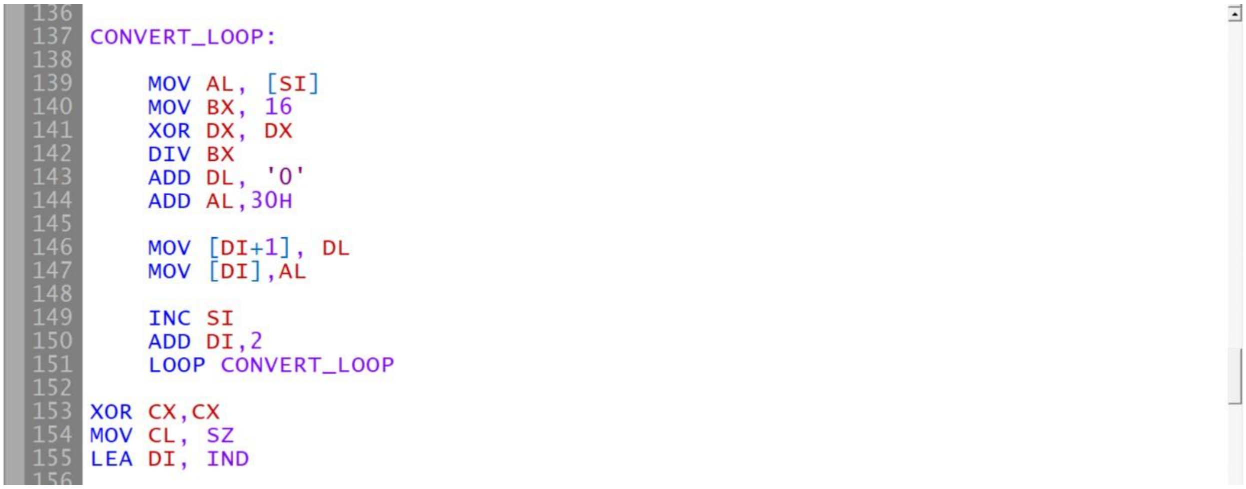
```
CONVERT:
    SHL AX, 4
    OR AL, BL      ; ADD NEW DIGIT TO AL
    INC SI        ; MOVE TO NEXT CHARACTER
    DEC CX
    JNZ MAIN_LOOP ;REPEAT FOR NEXT DIGIT

DONE:
    ;STORE IN ARRAY
    MOV [DI-2], AX
    POP CX
    DEC CX
    JNZ OUT_LOOP ;REPEAT FOR NEXT PLAYER

;--------------CODE----------

START: MOV BYTE PTR [FLG], 0
       LEA SI,TIM
       LEA DI,NUM
       XOR CX,CX
       MOV CL, SZ
       DEC CL
```



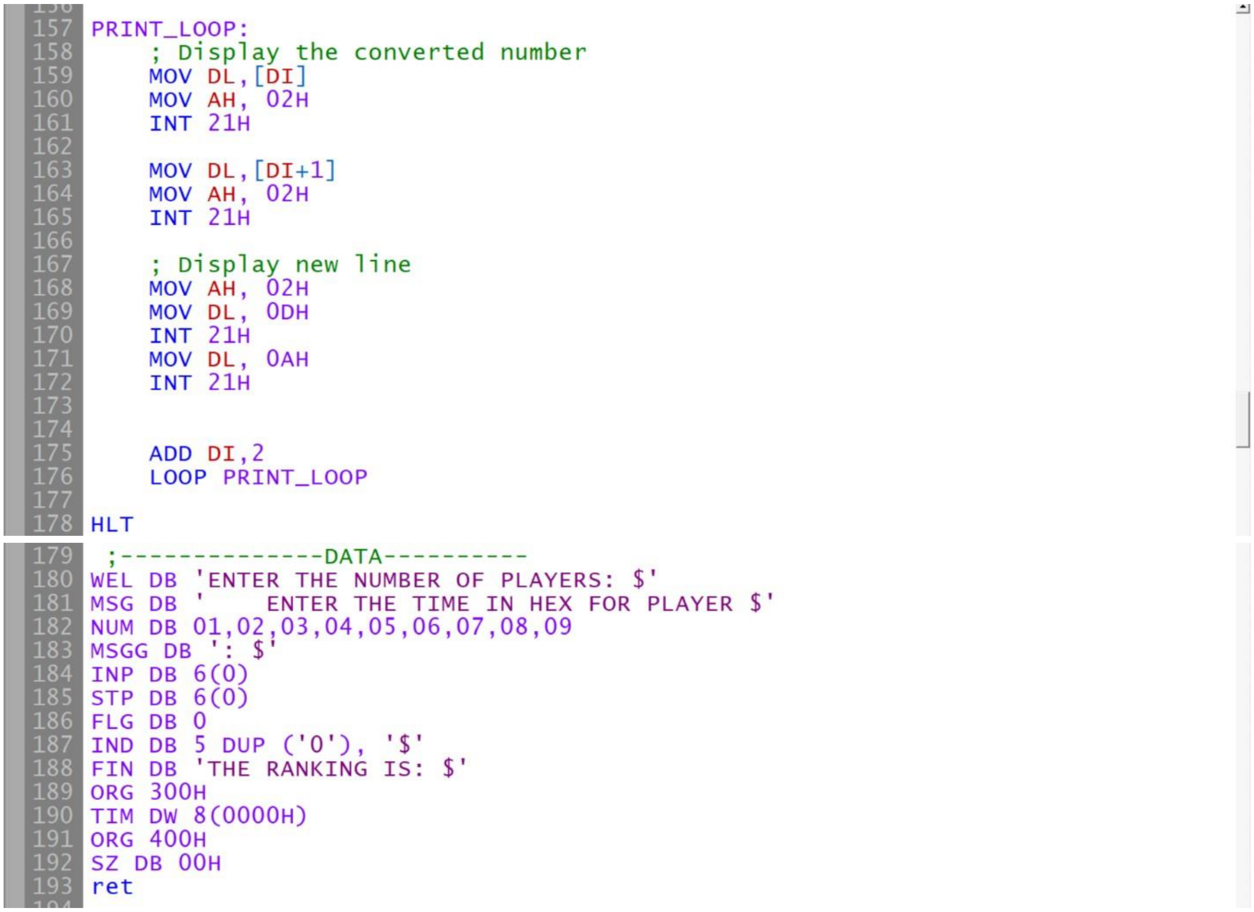
```
MAIN_LOOP1: MOV AX,[SI]
            MOV DX,[SI+2]

            MOV BL,[DI]
            MOV BH,[DI+1]

            CMP AX,DX
            JBE NO_SWAP
    SWAP:  MOV BYTE PTR [FLG], 1
           XCHG AX ,DX
           XCHG BL ,BH

           MOV [SI],AX
           MOV [SI+2],DX

           MOV [DI],BL
           MOV [DI+1],BH
NO_SWAP: ADD SI,2
         INC DI
         LOOP MAIN_LOOP1
         MOV BL,[FLG]
         DEC BL
         JZ START
```



```

;----------------------PRINT OUTPUTS------------------------

; Display new line
MOV AH, 02H
MOV DL, 0DH
INT 21H
MOV DL, 0AH
INT 21H
;Display message for ranking of players
MOV AH, 09H
LEA DX, FIN
INT 21H


LEA SI, NUM
LEA DI, IND
XOR CX,CX
MOV CL, SZ
XOR AX, AX
```

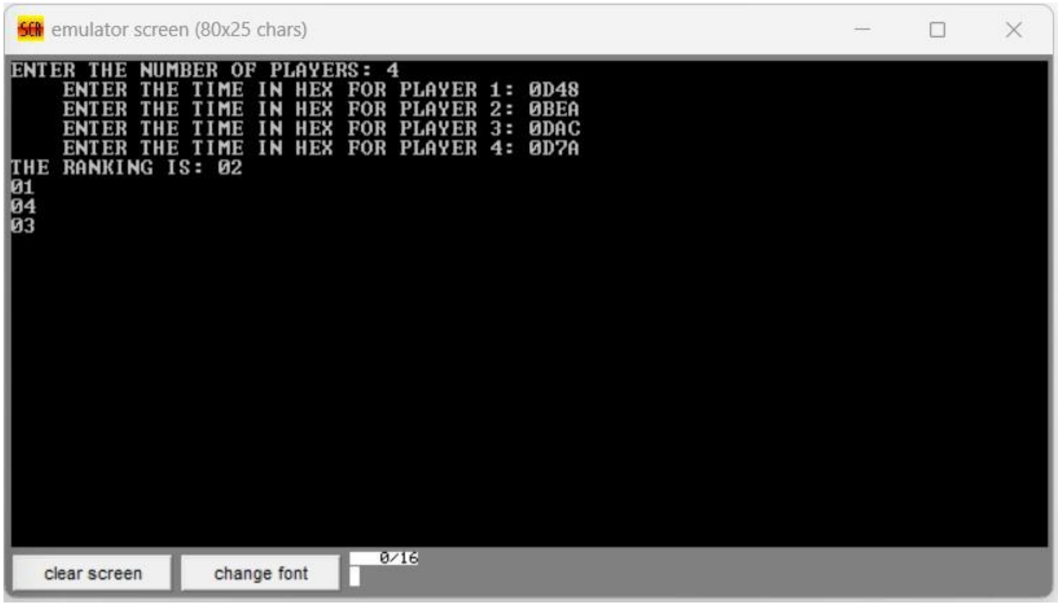


```

CONVERT_LOOP:

    MOV AL, [SI]
    MOV BX, 16
    XOR DX, DX
    DIV BX
    ADD DL, '0'
    ADD AL,30H

    MOV [DI+1], DL
    MOV [DI],AL

    INC SI
    ADD DI,2
    LOOP CONVERT_LOOP

XOR CX,CX
MOV CL, SZ
LEA DI, IND
```

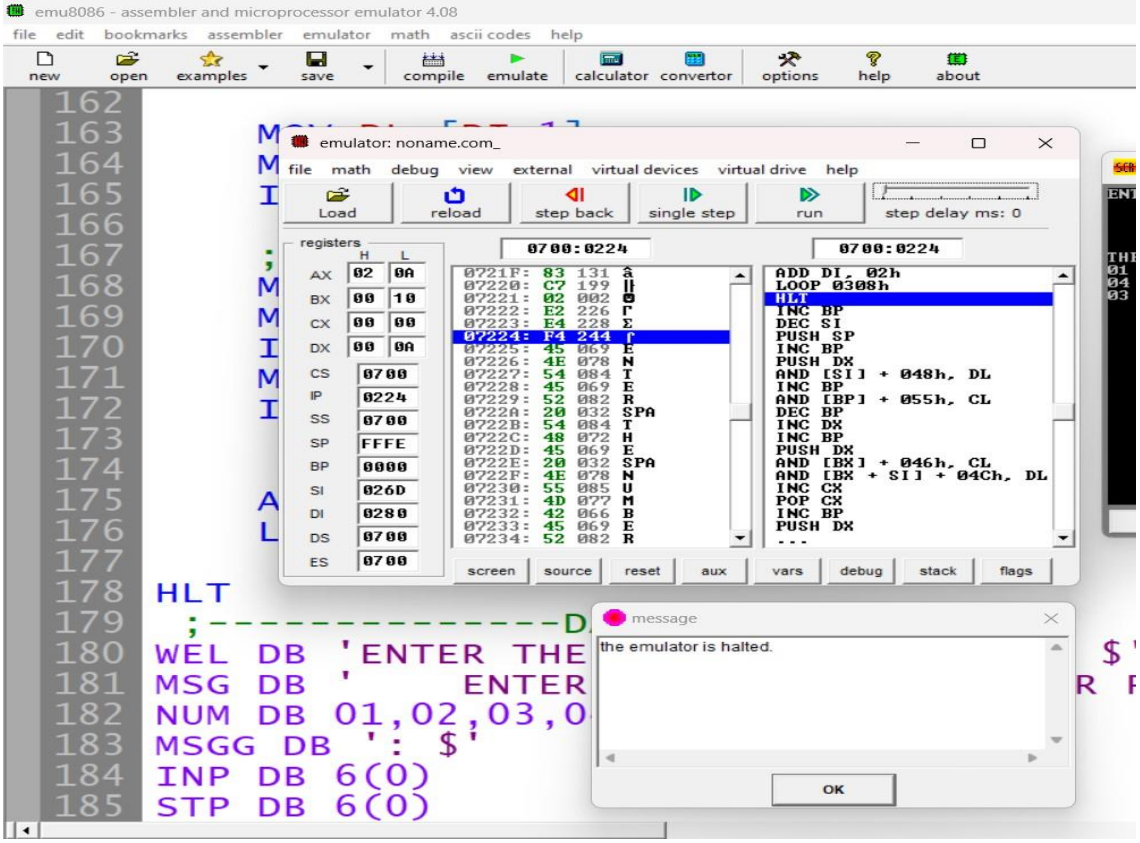


```
PRINT_LOOP:
    ; Display the converted number
    MOV DL,[DI]
    MOV AH, 02H
    INT 21H

    MOV DL,[DI+1]
    MOV AH, 02H
    INT 21H

    ; Display new line
    MOV AH, 02H
    MOV DL, 0DH
    INT 21H
    MOV DL, 0AH
    INT 21H


    ADD DI,2
    LOOP PRINT_LOOP

HLT
 ;--------------DATA----------
WEL DB 'ENTER THE NUMBER OF PLAYERS: $'
MSG DB '    ENTER THE TIME IN HEX FOR PLAYER $'
NUM DB 01,02,03,04,05,06,07,08,09
MSGG DB ': $'
INP DB 6(0)
STP DB 6(0)
FLG DB 0
IND DB 5 DUP ('0'), '$'
FIN DB 'THE RANKING IS: $'
ORG 300H
TIM DW 8(0000H)
ORG 400H
SZ DB 00H
ret
```

Results: first test case



```
ENTER THE NUMBER OF PLAYERS: 4
    ENTER THE TIME IN HEX FOR PLAYER 1: 0D48
    ENTER THE TIME IN HEX FOR PLAYER 2: 0BEA
    ENTER THE TIME IN HEX FOR PLAYER 3: 0DAC
    ENTER THE TIME IN HEX FOR PLAYER 4: 0D7A
THE RANKING IS: 02
01
04
03
```


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags
message
OK

```
HLT
 ;-------------D
WEL DB 'ENTER THE                                          $'
MSG DB '     ENTER                                       R P
NUM DB 01,02,03,0
MSGG DB ': $'
INP DB 6(0)
STP DB 6(0)
```

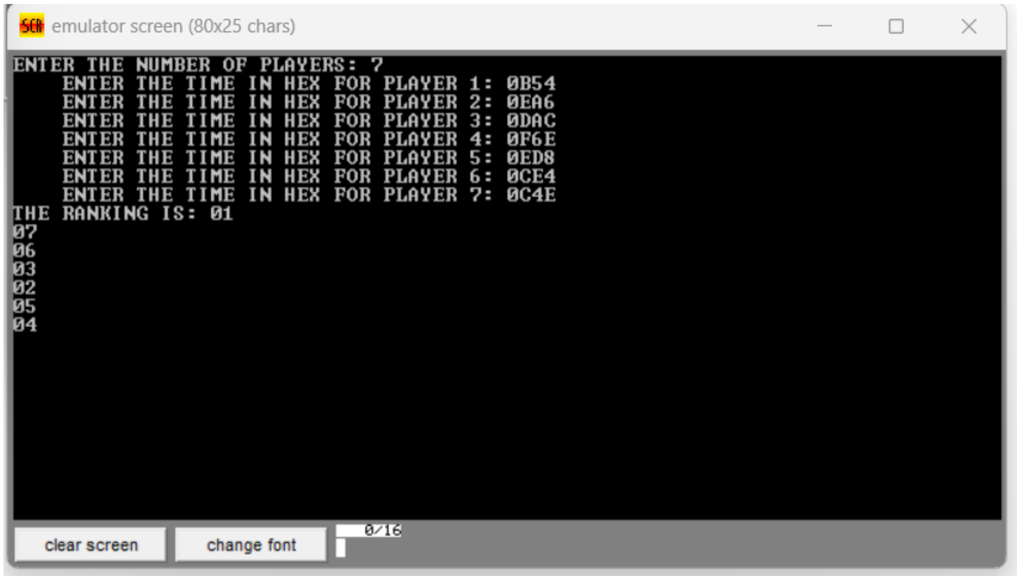
```
registers
      H    L
AX   02   0A
BX   00   10
CX   00   00
DX   00   0A
CS   0700
IP   0224
SS   0700
SP   FFFE
BP   0000
SI   026D
DI   0280
DS   0700
ES   0700
```

```
0700:0224
0721F: 83 131 â
07220: C7 199 ╟
07221: 02 002 ☻
07222: E2 226 Γ
07223: E4 228 Σ
07224: F4 244 ⌠
07225: 45 069 E
07226: 4E 078 N
07227: 54 084 T
07228: 45 069 E
07229: 52 082 R
0722A: 20 032 SPA
0722B: 54 084 T
0722C: 48 072 H
0722D: 45 069 E
0722E: 20 032 SPA
0722F: 4E 078 N
07230: 55 085 U
07231: 4D 077 M
07232: 42 066 B
07233: 45 069 E
07234: 52 082 R
```

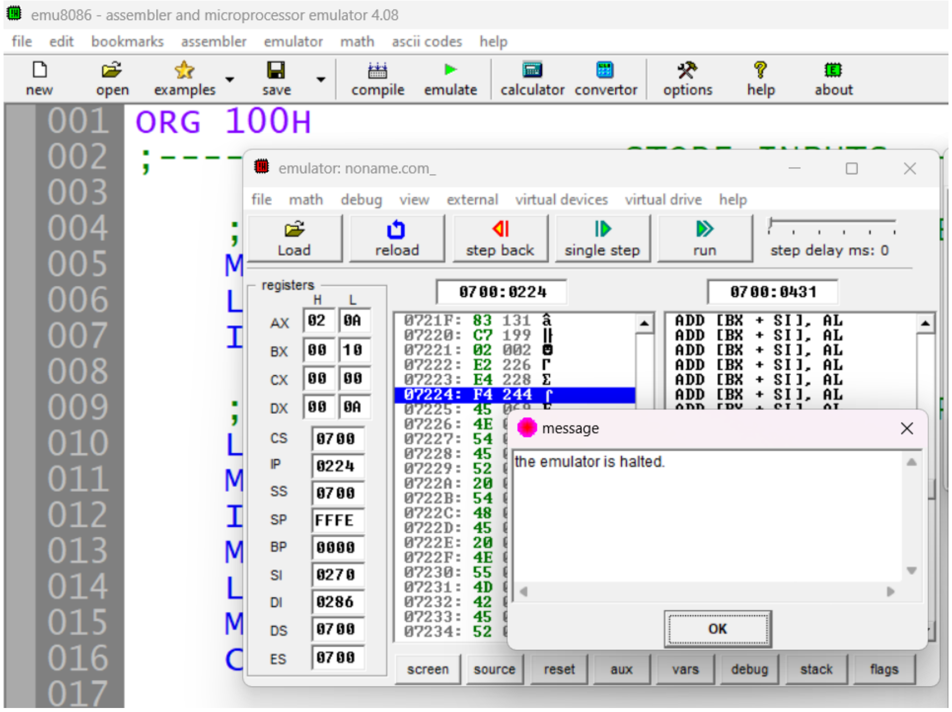
```
0700:0224
ADD DI, 02h
LOOP 0308h
HLT
INC BP
DEC SI
PUSH SP
INC BP
PUSH DX
AND [SI] + 048h, DL
INC BP
AND [BP] + 055h, CL
DEC BP
INC DX
INC BP
PUSH DX
AND [BX] + 046h, CL
AND [BX + SI] + 04Ch, DL
INC CX
POP CX
INC BP
PUSH DX
...
```

the emulator is halted.

SECOND TEST CASE



```
ENTER THE NUMBER OF PLAYERS: 7
    ENTER THE TIME IN HEX FOR PLAYER 1: 0B54
    ENTER THE TIME IN HEX FOR PLAYER 2: 0EA6
    ENTER THE TIME IN HEX FOR PLAYER 3: 0DAC
    ENTER THE TIME IN HEX FOR PLAYER 4: 0F6E
    ENTER THE TIME IN HEX FOR PLAYER 5: 0ED8
    ENTER THE TIME IN HEX FOR PLAYER 6: 0CE4
    ENTER THE TIME IN HEX FOR PLAYER 7: 0C4E
THE RANKING IS: 01
07
06
03
02
05
04
```


emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load reload step back single step run step delay ms: 0
screen source reset aux vars debug stack flags

```
ORG 100H
```

registers	H	L
AX	02	0A
BX	00	10
CX	00	00
DX	00	0A
CS	0700	
IP	0224	
SS	0700	
SP	FFFE	
BP	0000	
SI	0270	
DI	0286	
DS	0700	
ES	0700	

```
0700:0224

0721F: 83 131 â
07220: C7 199 ╟
07221: 02 002 ☻
07222: E2 226 Γ
07223: E4 228 Σ
07224: F4 244 ⌠
07225: 45
07226: 4E
07227: 54
07228: 45
07229: 52
0722A: 20
0722B: 54
0722C: 48
0722D: 45
0722E: 20
0722F: 4E
07230: 55
07231: 4D
07232: 42
07233: 45
07234: 52
```

```
0700:0431

ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
ADD [BX + SI], AL
```

message

the emulator is halted.

OK

Summary

- **Functionality**: This program prompts the user for the number of players and their respective times in hexadecimal. It then sorts these times and outputs the sorted times in ranking order.
- **Input/Output**: Uses DOS interrupts (INT 21H) for input and output. Takes the number of players as input and then takes a series of times in hexadecimal format, sorts them, and outputs the sorted result.

Code Breakdown

1. Setup and Input Handling

- **ORG 100H**: Defines the starting offset for the code.
- **MOV AH, 09H, LEA DX, WEL, INT 21H**: Display a message prompting for the number of players.
- **MOV AH, 0AH, LEA DX, STP, INT 21H**: Reads an ASCII string of characters into a specified buffer (defined in the data section).
- **MOV SZ[0], CL**: Stores the converted number of players into a byte array.

2. Reading Player Times

- The code reads the time for each player in a loop (**OUT_LOOP**). It displays a prompt message and reads the user input.
- **ASCII to Hexadecimal Conversion**: This part of the code converts ASCII characters into hexadecimal numbers by subtracting **'0'** from the character code. If it goes beyond **'9'**, it adjusts by subtracting 7, considering it a hexadecimal letter (e.g., **'A' - '0' + 10 = 10**).

3. Sorting the Times

- The sorting algorithm resembles a simple bubble sort:
    - The loop iterates through the array of times, comparing pairs of times.
    - If a time is less than the following time, it swaps the values (both in **TIM** array and **NUM** array, keeping track of which player has what time).
    - The loop continues until no swaps are needed.

4. Outputting the Result

- After sorting, the program displays the times in ascending order.
- **MOV AH, 02H, MOV DL, 0DH/0AH, INT 21H**: This combination prints a newline on the screen.
- **LEA DI, IND, MOV CL, SZ**: These instructions setup the output array with correct values to be displayed.

5. Data Section

- **ORG 300H**: Defines the offset for the **TIM** array.
- **TIM DW 8(0000H)**: Array that stores the hexadecimal times for the players.
- **SZ DB 00H**: A variable to store the number of players.

Conclusion

This code involves user interaction, data sorting, and conversion, demonstrating a combination of skills in assembly language programming. It uses loop structures, conditional statements, and direct memory manipulation, illustrating the core elements of low-level programming.