ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

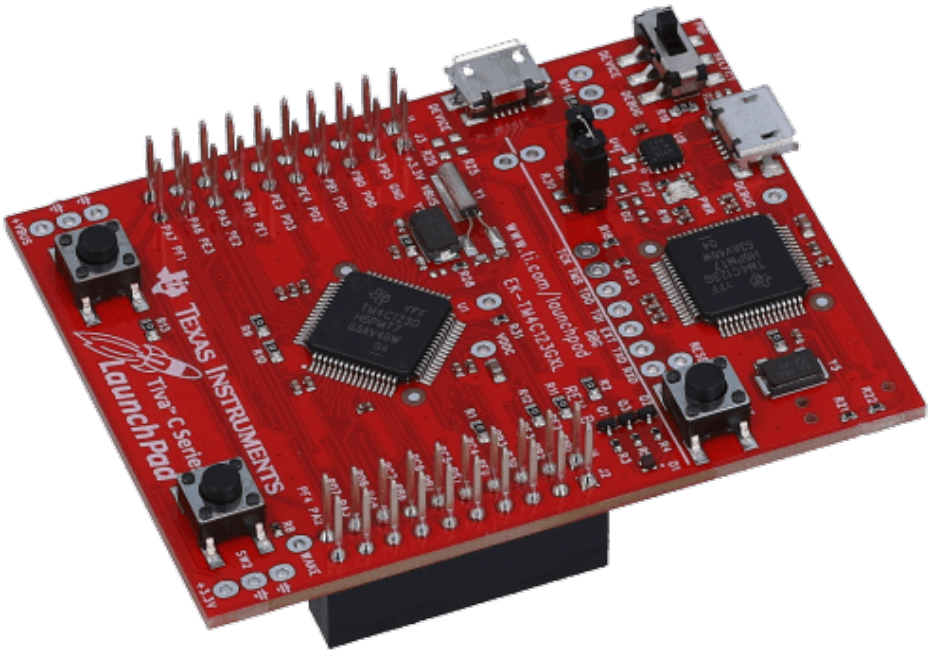
EE447- Introduction to Microprocessors Laboratory With Assembly Programming

(Preliminary Work 0)

EXPERIMENTAL WORK NO: 0

1st Group Member: Uğur SAMANCI - 2398915

2nd Group Member: Barış GÜZEL - 2304764



Question 1) Build, run and understand Practice Lab.s

- First part of the preliminary work, we analyzed Practibe Lab.s

```
13  ;*****************************************************************
14  ;SYMBOL      DIRECTIVE    VALUE           COMMENT
15  FIRST       EQU          0x20000400
16  CONST       EQU          0x20
17  ;*****************************************************************
18  ; Program section
```

Figure 1: Symbol's and their values

On line 15: we add a value of 0x20000400 to a symbol called FIRST to use it later.

On line 16: we add a value of 0x20 to a symbol called CONST to use it later.

```
__main      PROC
            LDR         R1,=FIRST       ; Initialize registers
            MOV         R0,#0x00        ;
            LDR         R2,=CONST       ;
```

Figure 2: Initialized register values

On line 25, 26, & 27 we initialized R0, R1, & R2 one can see the values on Figure 3.

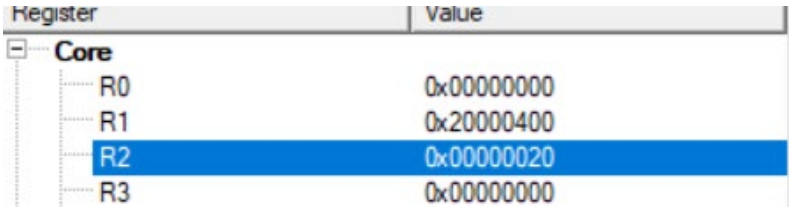


Figure 3: Register values screenshot

Before entering loop 1 one can see relative adress and their contents in Figure 4. & Figure 5.

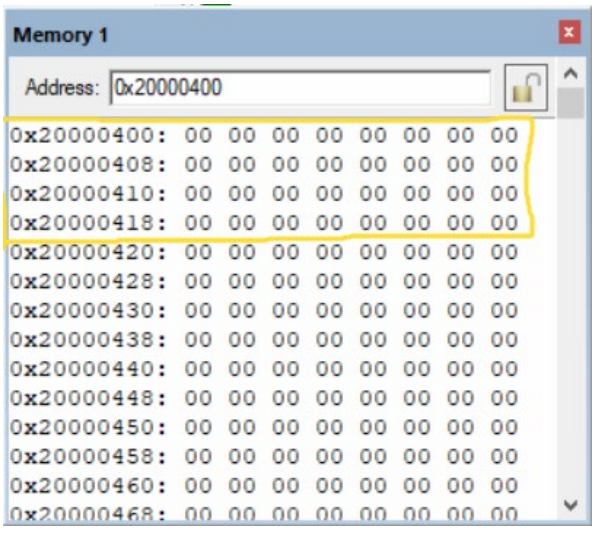


Figure 4: Specified adress locations:

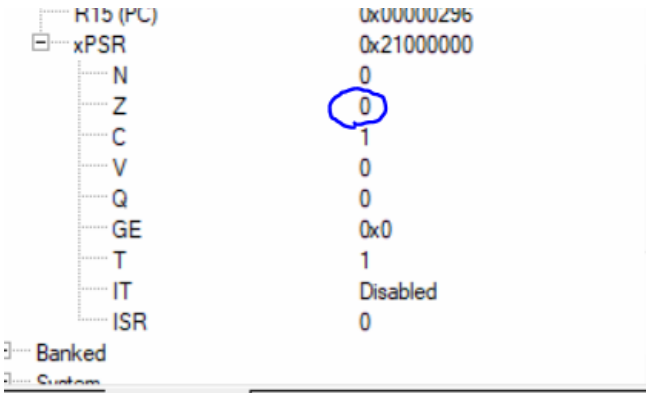


Figure 5: Zero bit shows 0

After the first loop ended in other Word when Z bit set to 1 one can see relative adress and their contents in Figure 6.

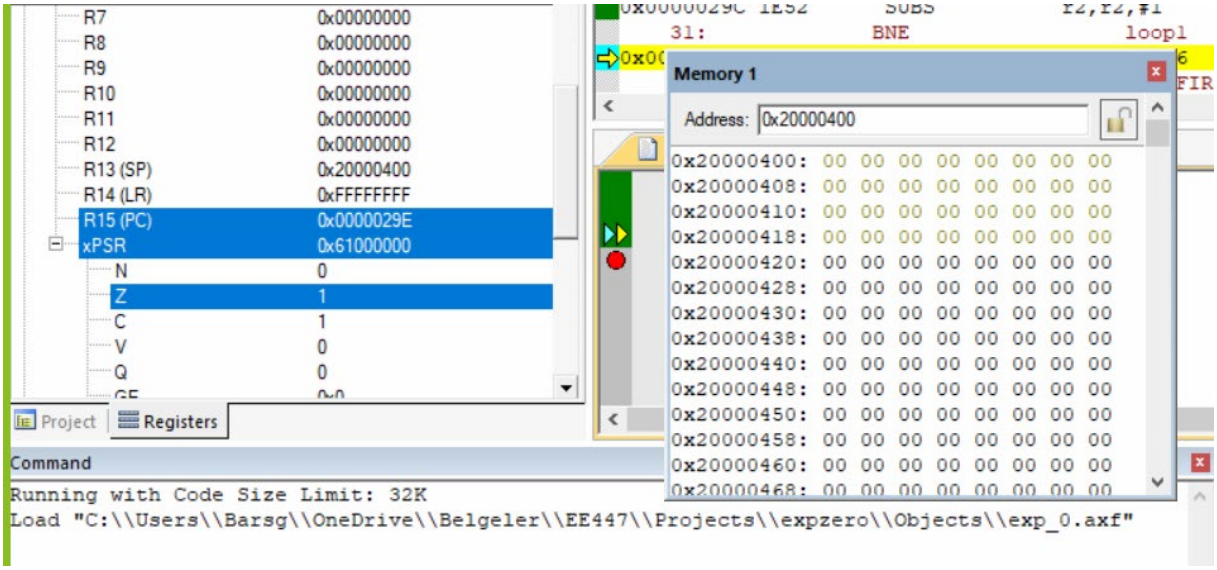


Figure 6: Zero bit setted to 1 and specified memory locations set to 0

After the second loop ended in other Word when Z bit set to 1 ended.

- In this part, we understood that R2 is used to set loop's iteration number as 32 times which is equal to 0x20 in hexadecimal. After the first loop we reset the adress of the R1 register so that we can store numbers from 0 to 1F in address interval of 0x20000400-0x2000041F.
- One can see change in the memory interval in Figure 7.

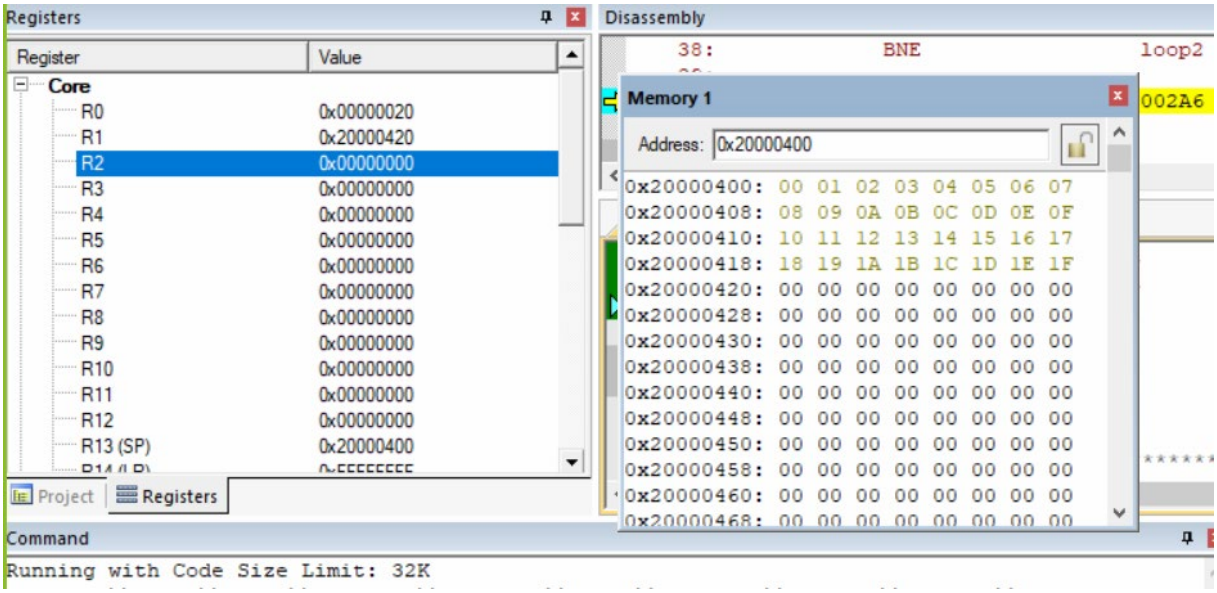


Figure 7: Specified memory locations filled with consequitive numbers 00-1F

Question 2) Build, run and understand Practice Lab.s

- In this part we simply stored and copied an address interval from 0x20000400 to 0x2000040F then we copied to address interval from 0x20000410 to 0x2000041F.
- On Figure 8. One can see copying table command on Termite 3.4 and relatives lines.

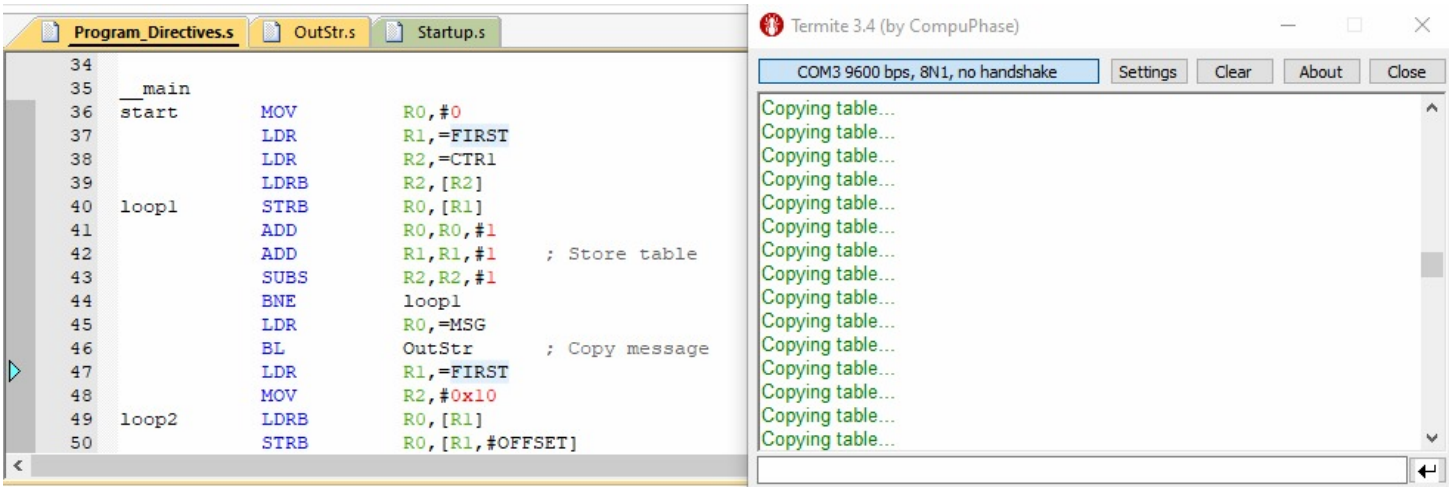


Figure 8: Showing the relative lines and Termite connection screen

- On Figure 9. One can see the relative copy paste job in memory address location.

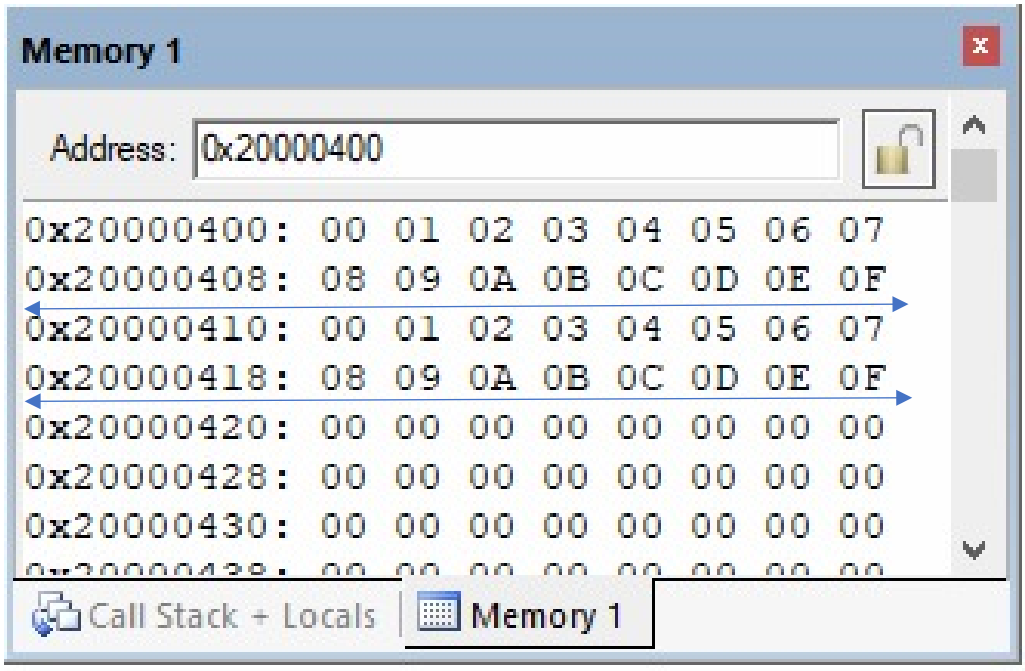


Figure 9: Relative address locations

Question 3) Make the following modifications on Program Directives.s.

- First of all we changed the symbol FIRST value from 0x20000400 to 0x20000700 to create our new starting address:
- Then we changed our constant CONST from 0x10 to 0x22 in hexademical because note that from 0 to 10 we have 17 value in decimal multiply it by two we get 34 which is 0x22.
- Then we changed CTR1 DCB 0x10 to CTR1 DCB 0x11
- Finally in both loop1 and loop2 we add 2 lines respectively;

```
ADD     R1,R1,#1

STRB    R0,[R1]
```

- We added two lines above because we want everything twice
- One can see the memory in Figure 10.

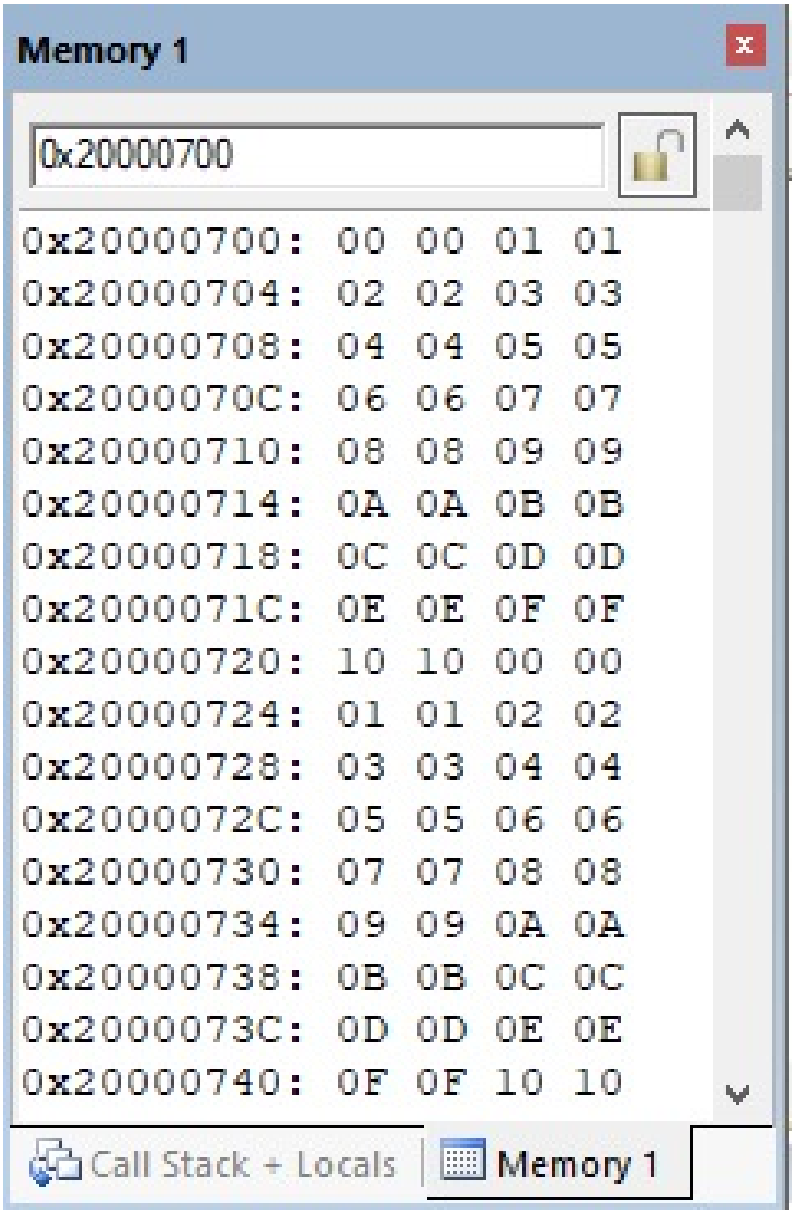


Figure 10: Changed table in part 3

```
;*********************
; Program_Directives.s
; Copies the table from one location
; to another memory location.
; Directives and Addressing modes are
; explained with this program.
;*********************
;*********************
; EQU Directives
; These directives do not allocate memory
;*********************
;LABEL     DIRECTIVE   VALUE      COMMENT
OFFSET     EQU         0x22                                                ;We
changed from 0x10 to 0x22
FIRST      EQU         0x20000700  ; New starting address location
;*********************
; Directives - This Data Section is part of the code
; It is in the read only section  so values cannot be changed.
;*********************
;LABEL     DIRECTIVE   VALUE      COMMENT
           AREA        sdata, DATA, READONLY
           THUMB
CTR1       DCB         0x11                                                ;We
changed from 0x10 to 0x11
MSG        DCB         "Copying table..."
           DCB         0x0D
           DCB         0x04
;*********************
; Program section
;*********************
;LABEL     DIRECTIVE   VALUE      COMMENT
           AREA        main, READONLY, CODE
           THUMB
           EXTERN      OutStr     ; Reference external subroutine
           EXPORT      __main     ; Make available

__main
start      MOV         R0,#0
           LDR         R1,=FIRST                                           ;New address
0x20000700
           LDR         R2,=CTR1
           LDRB        R2,[R2]
loop1      STRB        R0,[R1]

                                                                           ADD

                                                                           R1,R1,#1
```

```
                                                                    ;Since we
want every value twice, we added this line

                                                                    STRB
R0,[R1]                                                                 ;Since
we want every value twice, we added this line
        ADD     R0,R0,#1
        ADD     R1,R1,#1    ; Store table
        SUBS    R2,R2,#1
        BNE     loop1
        LDR     R0,=MSG
        BL      OutStr      ; Copy message
        LDR     R1,=FIRST
        MOV     R2,#0x11
loop2   LDRB    R0,[R1]
        STRB    R0,[R1,#OFFSET]
        ADD     R1,R1,#1    ; Copy table

                                                                    STRB
R0,[R1,#OFFSET] ; Since we want every value twice, we added this line
        ADD     R1,R1,#1    ;Since we want every value twice, we added this line
        SUBS    R2,R2,#1
        BNE     loop2
        B       start
;*********************
; End of the program  section
;*********************
;LABEL   DIRECTIVE   VALUE               COMMENT
        ALIGN
        END
```

Question 4) Write the program given in 1.10. You will have to add InChar.s, OutChar.s to your project folder.

- This part is simple input and output part whatever we give as input we take it as output example can be seen in Figure 11. Added code can be found after the figure below.

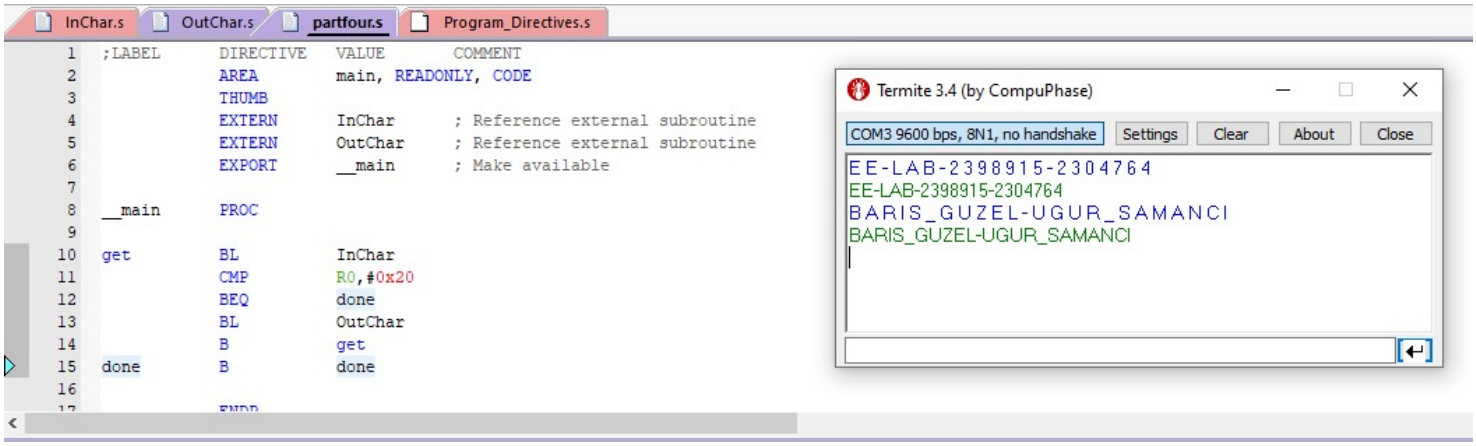


Figure 11: Example of code in 1.10

```
;LABEL          DIRECTIVE       VALUE           COMMENT
                        AREA    main, READONLY, CODE
                        THUMB
                        EXTERN          InChar          ; Reference external subroutine
                        EXTERN          OutChar                 ; Reference external subroutine
                        EXPORT          __main          ; Make available

__main          PROC

get             BL                      InChar
                        CMP                     R0,#0x20
                        BEQ                     done
                        BL                      OutChar
                        B                       get
done            B                       done

                        ENDP


;*********************
; End of the program  section
;*********************
;LABEL     DIRECTIVE     VALUE                   COMMENT
                        ALIGN
                        END
```

Question 5) Build, run and understand Program Directives.c. You have to add OutStr.s to your project.

- We added Program Directives.c file and we get exactly same result with .s file

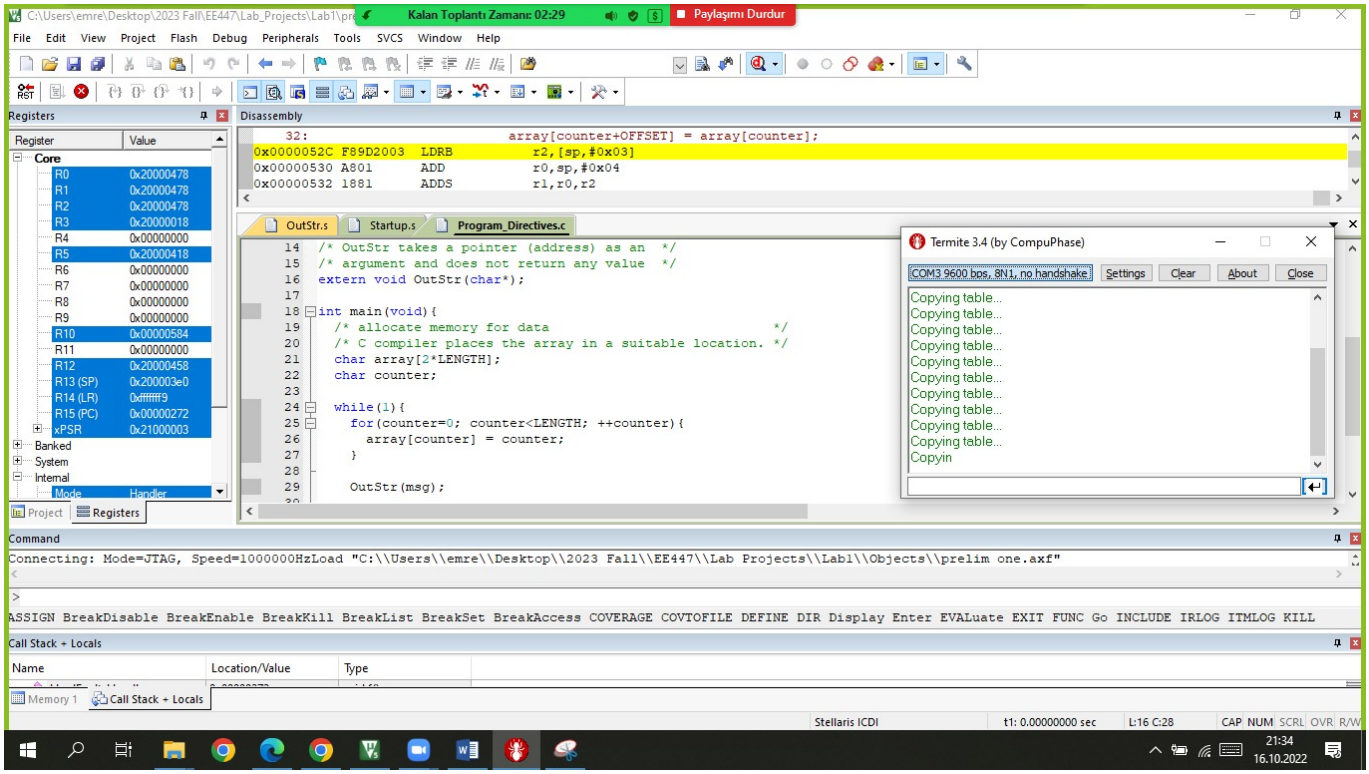


Figure 12: Screenshot for part five

Question 6) Rewrite the program given in 1.10 in C language. You will have to add InChar.s, OutChar.s
to your project.