Middle East Technical University

Electrical and Electronics Engineering Department

EE447
Introduction to Microprocessors

Preliminary Report Laboratory 1

Fazlı Oğulcan Baytimur

ID: 2231389

13/12/2021

Chapter 1

Preliminary Work

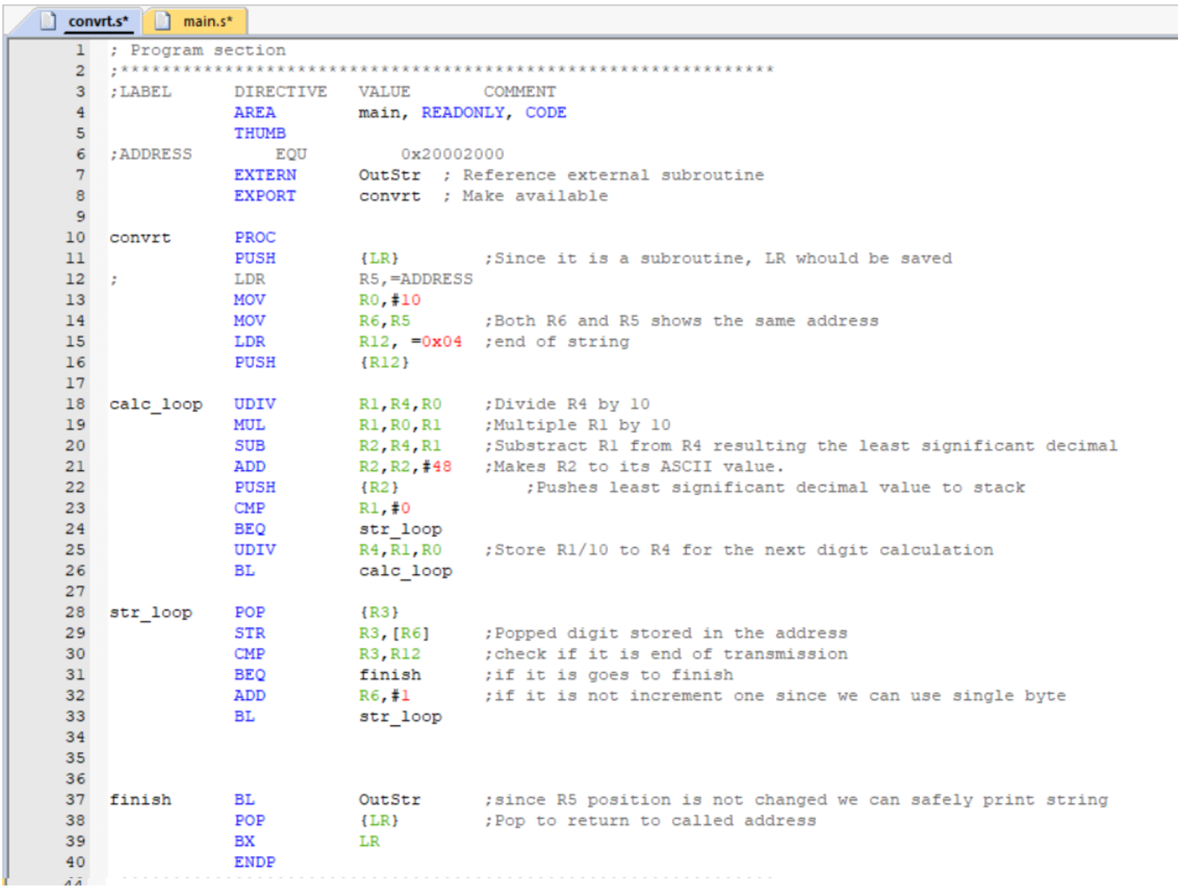
1.1 Question 1

Algorithm 1 An algorithm for hex-to-dec converter

```
Require: $R4 \leftarrow Hex$
  Calculation Loop
  PUSH LR to the Stack
  $R1 \leftarrow \frac{R4}{10}$
  $R1 \leftarrow R1 \times 10$
  $R2 \leftarrow R4 - R1$
  PUSH $R2$ to the Stack
  if $R1 = 0$ then
      Go to String Loop
  end if
  $R4 \leftarrow \frac{R1}{10}$
  Go back to Calculation Loop

  String Loop
  POP $R3$
  $[R6] \leftarrow R3$
  if $R3 = 0x04$ then
      Go to Finish
  end if
  Increment $R6$ by 1
  Go back to String Loop

  Finish
  Use OutStr Subroutine
  POP LR
  Branch to LR
```

The general concept is trivial that, to find a number in a different base, the desired number is divided with the desired base many time. In the end, the remainders builds up the number in that base. The same approach has been followed as it can be seen in

Algorithm 1. Since the terminal write the values in ASCII form, the remainders stored with increments of 48 since "0" start at 48 in ASCII. All the pseudo code can also be seen in 1. Moreover, the code version of this algorithm can be seen in Figure 1.1.

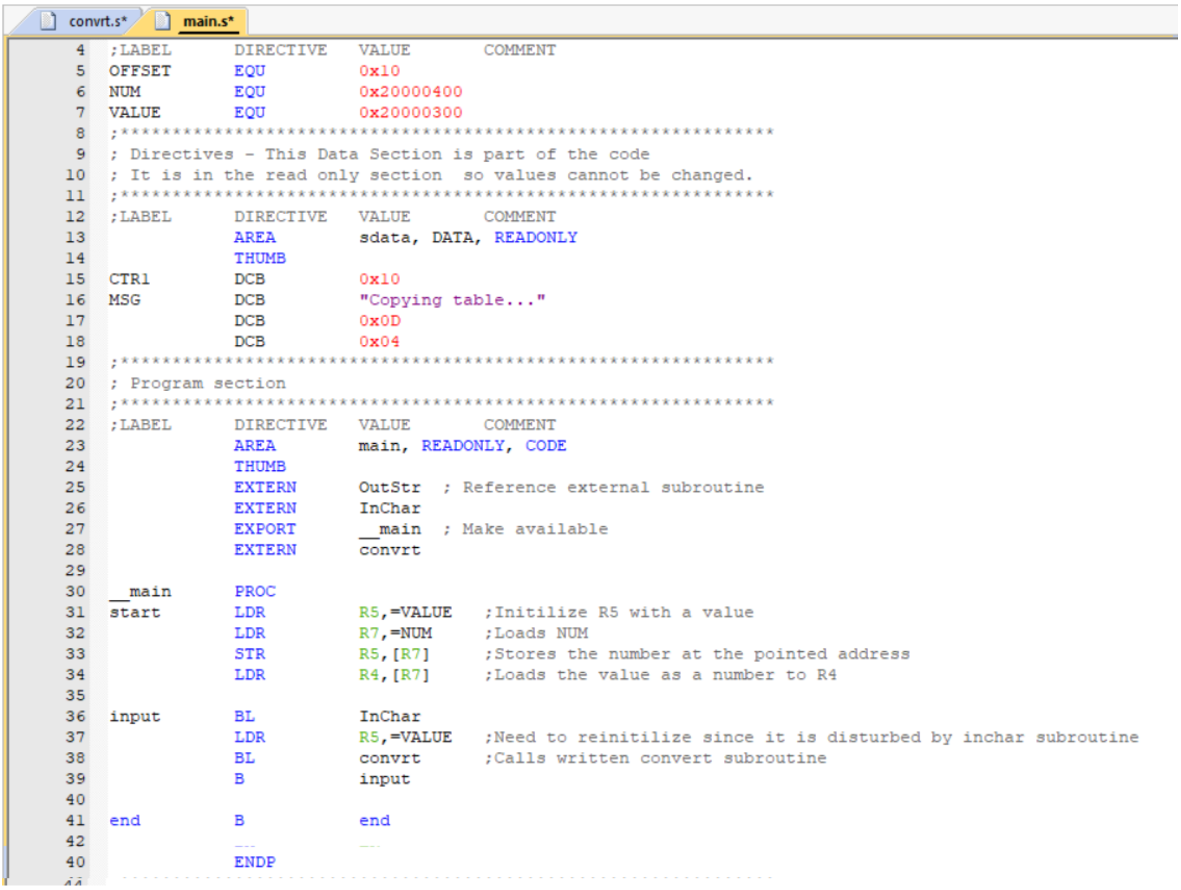


```
; Program section
;***************************************************************
;LABEL      DIRECTIVE   VALUE       COMMENT
            AREA        main, READONLY, CODE
            THUMB
;ADDRESS        EQU         0x20002000
            EXTERN      OutStr  ; Reference external subroutine
            EXPORT      convrt  ; Make available

convrt      PROC
            PUSH        {LR}        ;Since it is a subroutine, LR whould be saved
;           LDR         R5,=ADDRESS
            MOV         R0,#10
            MOV         R6,R5       ;Both R6 and R5 shows the same address
            LDR         R12, =0x04  ;end of string
            PUSH        {R12}

calc_loop   UDIV        R1,R4,R0    ;Divide R4 by 10
            MUL         R1,R0,R1    ;Multiple R1 by 10
            SUB         R2,R4,R1    ;Substract R1 from R4 resulting the least significant decimal
            ADD         R2,R2,#48   ;Makes R2 to its ASCII value.
            PUSH        {R2}            ;Pushes least significant decimal value to stack
            CMP         R1,#0
            BEQ         str_loop
            UDIV        R4,R1,R0    ;Store R1/10 to R4 for the next digit calculation
            BL          calc_loop

str_loop    POP         {R3}
            STR         R3,[R6]     ;Popped digit stored in the address
            CMP         R3,R12      ;check if it is end of transmission
            BEQ         finish      ;if it is goes to finish
            ADD         R6,#1       ;if it is not increment one since we can use single byte
            BL          str_loop



finish      BL          OutStr      ;since R5 position is not changed we can safely print string
            POP         {LR}        ;Pop to return to called address
            BX          LR
            ENDP
```

Figure 1.1: Code of the convrt subroutine in KEIL

1.2 Question 2

The convrt subroutine should take the parameter of the hex number which it is going to calculate. The hex number can straight-forwardly pass through registers of which the ones used in convrt subroutine. The code part can be seen in Figure 1.2.



```
;LABEL      DIRECTIVE   VALUE       COMMENT
OFFSET      EQU         0x10
NUM         EQU         0x20000400
VALUE       EQU         0x20000300
;**************************************************************
; 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         0x10
MSG         DCB         "Copying table..."
            DCB         0x0D
            DCB         0x04
;**************************************************************
; Program section
;**************************************************************
;LABEL      DIRECTIVE   VALUE       COMMENT
            AREA        main, READONLY, CODE
            THUMB
            EXTERN      OutStr  ; Reference external subroutine
            EXTERN      InChar
            EXPORT      __main  ; Make available
            EXTERN      convrt

__main      PROC
start       LDR         R5,=VALUE   ;Initilize R5 with a value
            LDR         R7,=NUM     ;Loads NUM
            STR         R5,[R7]     ;Stores the number at the pointed address
            LDR         R4,[R7]     ;Loads the value as a number to R4

input       BL          InChar
            LDR         R5,=VALUE   ;Need to reinitilize since it is disturbed by inchar subroutine
            BL          convrt      ;Calls written convert subroutine
            B           input

end         B           end

            ENDP
```

Figure 1.2: Code of the convertion program using convrt subroutine.

1.3 Question 3

In this part, the user enters a n value to make a binary search between 1-2n̂. The program stops getting inputs until it sees '/' and warns user if n>32. As user tells the program upper, down or correct with the corresponding letters responding to program's guess'. The program changes the lower and upper boundaries in respect to the user's answers. The flowchart of this program and codes of the subroutines can be seen in the Figures 1.3 1.4 1.5 1.6 respectedly.

1.3.1 Flowchart

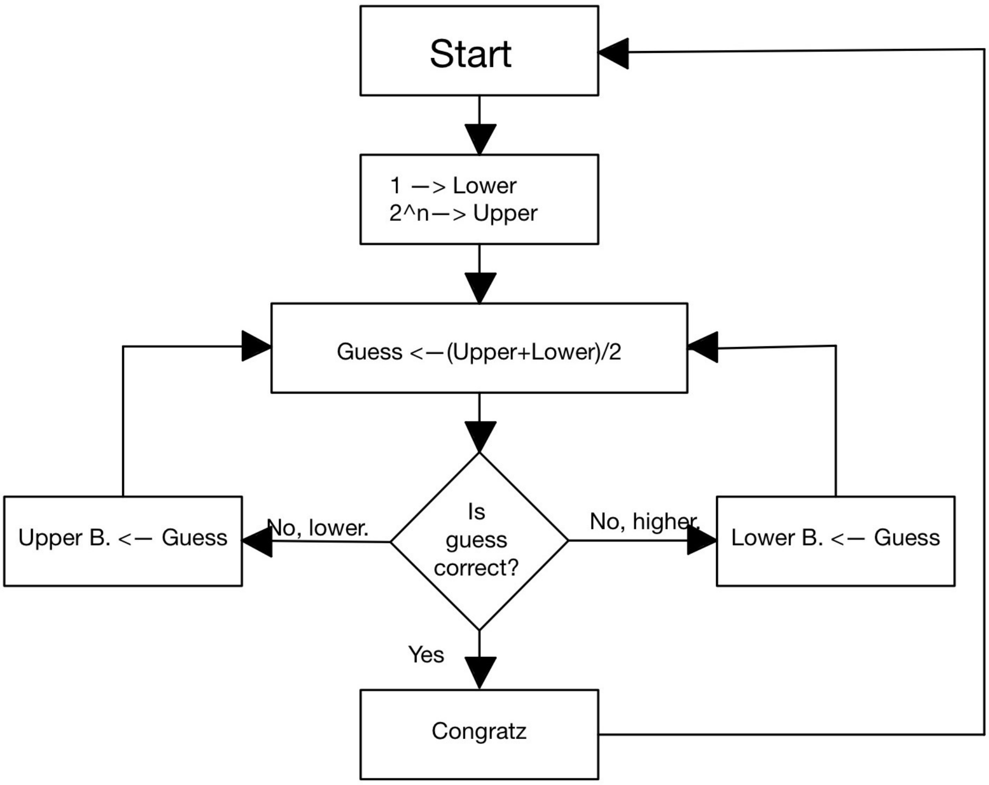
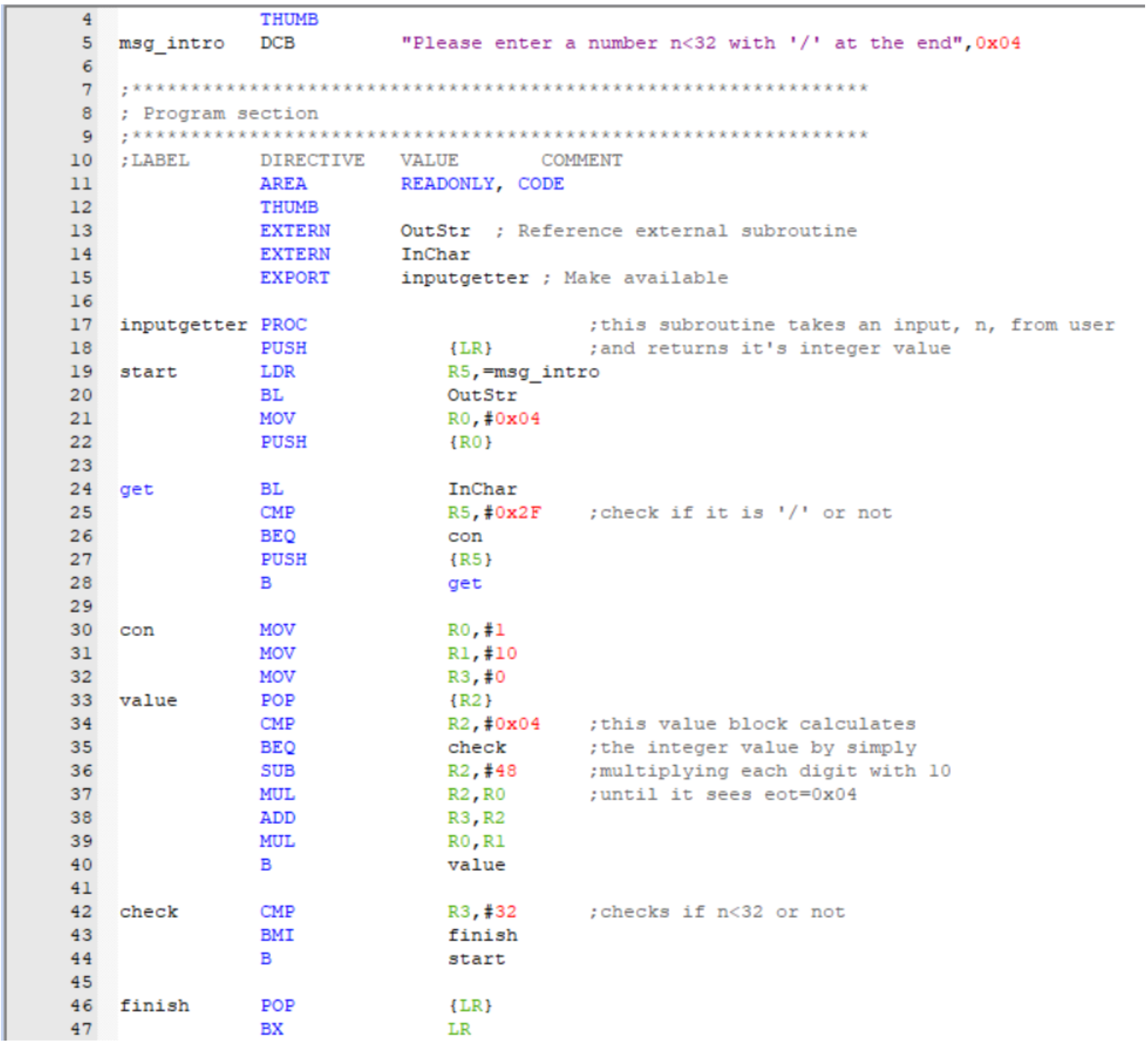


Figure 1.3: Flowcart of algorithm in question 3.

1.3.2 Inputgetter Subroutine

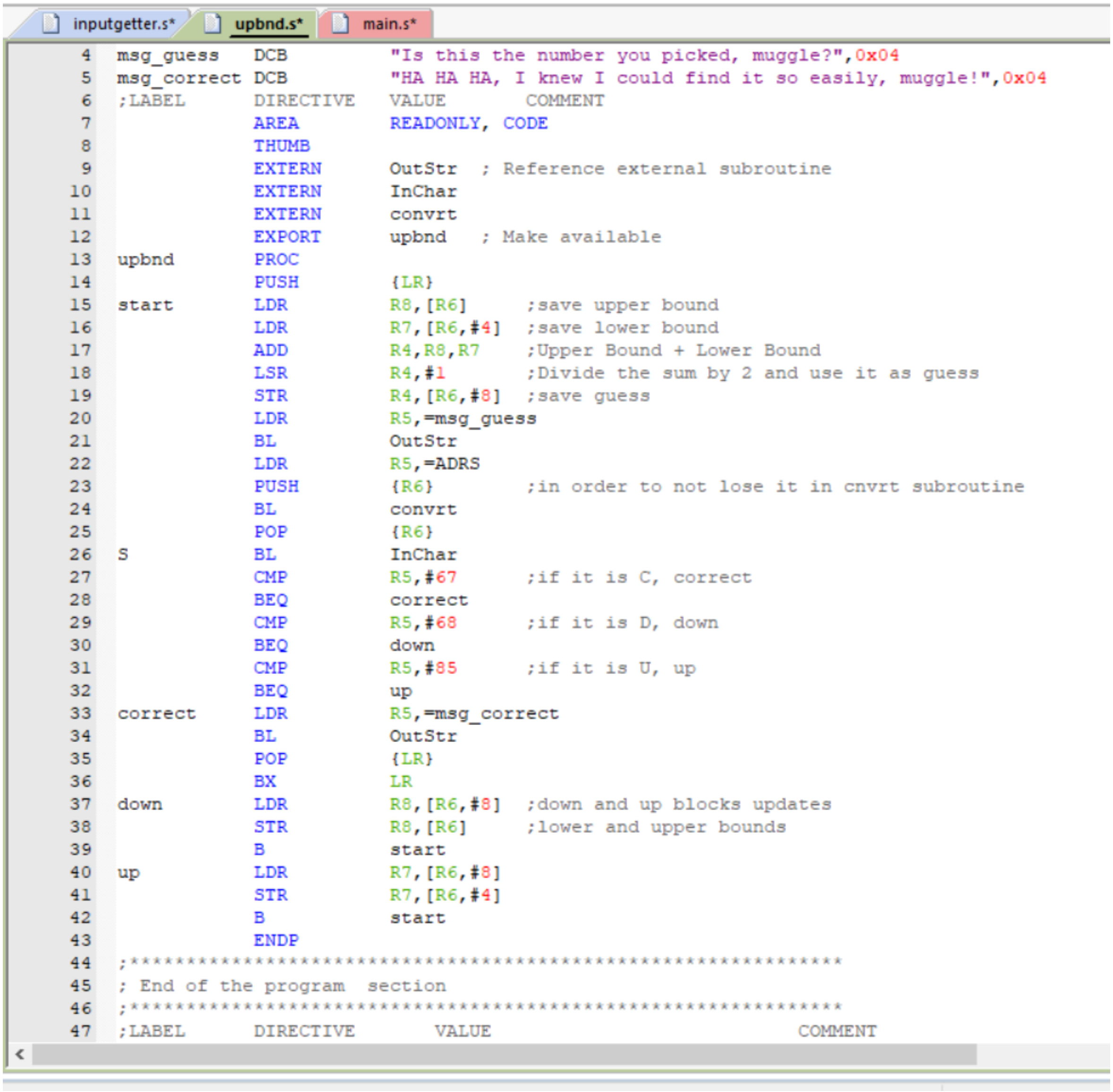
In order to make main cleaner, inputgetter subroutine is used. This subroutine takes the value n from the user and it calculates it's integer form and gives back to system.



```
            THUMB
msg_intro   DCB         "Please enter a number n<32 with '/' at the end",0x04

;****************************************************************
; Program section
;****************************************************************
;LABEL      DIRECTIVE   VALUE       COMMENT
            AREA        READONLY, CODE
            THUMB
            EXTERN      OutStr  ; Reference external subroutine
            EXTERN      InChar
            EXPORT      inputgetter ; Make available

inputgetter PROC                    ;this subroutine takes an input, n, from user
            PUSH            {LR}        ;and returns it's integer value
start       LDR             R5,=msg_intro
            BL              OutStr
            MOV             R0,#0x04
            PUSH            {R0}

get         BL              InChar
            CMP             R5,#0x2F    ;check if it is '/' or not
            BEQ             con
            PUSH            {R5}
            B               get

con         MOV             R0,#1
            MOV             R1,#10
            MOV             R3,#0
value       POP             {R2}
            CMP             R2,#0x04    ;this value block calculates
            BEQ             check       ;the integer value by simply
            SUB             R2,#48      ;multiplying each digit with 10
            MUL             R2,R0       ;until it sees eot=0x04
            ADD             R3,R2
            MUL             R0,R1
            B               value

check       CMP             R3,#32      ;checks if n<32 or not
            BMI             finish
            B               start

finish      POP             {LR}
            BX              LR
```

Figure 1.4: Inputgetter subroutine code

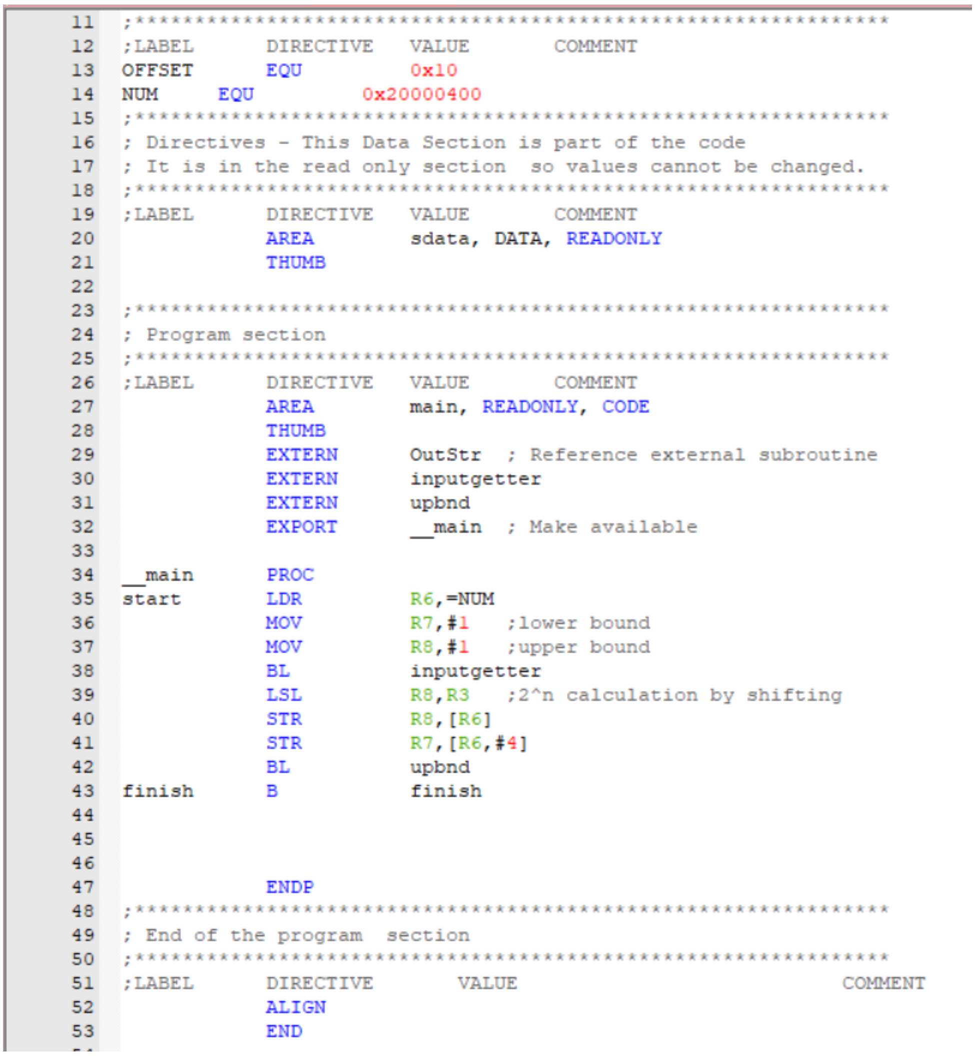
1.3.3 UPBND



```
msg_guess    DCB          "Is this the number you picked, muggle?",0x04
msg_correct  DCB          "HA HA HA, I knew I could find it so easily, muggle!",0x04
;LABEL       DIRECTIVE    VALUE       COMMENT
             AREA         READONLY, CODE
             THUMB
             EXTERN       OutStr  ; Reference external subroutine
             EXTERN       InChar
             EXTERN       convrt
             EXPORT       upbnd   ; Make available
upbnd        PROC
             PUSH         {LR}
start        LDR          R8,[R6]     ;save upper bound
             LDR          R7,[R6,#4]  ;save lower bound
             ADD          R4,R8,R7    ;Upper Bound + Lower Bound
             LSR          R4,#1       ;Divide the sum by 2 and use it as guess
             STR          R4,[R6,#8]  ;save guess
             LDR          R5,=msg_guess
             BL           OutStr
             LDR          R5,=ADRS
             PUSH         {R6}        ;in order to not lose it in cnvrt subroutine
             BL           convrt
             POP          {R6}
S            BL           InChar
             CMP          R5,#67      ;if it is C, correct
             BEQ          correct
             CMP          R5,#68      ;if it is D, down
             BEQ          down
             CMP          R5,#85      ;if it is U, up
             BEQ          up
correct      LDR          R5,=msg_correct
             BL           OutStr
             POP          {LR}
             BX           LR
down         LDR          R8,[R6,#8]  ;down and up blocks updates
             STR          R8,[R6]     ;lower and upper bounds
             B            start
up           LDR          R7,[R6,#8]
             STR          R7,[R6,#4]
             B            start
             ENDP
;*****************************************************************
; End of the program  section
;*****************************************************************
;LABEL       DIRECTIVE        VALUE                             COMMENT
```

Figure 1.5: UPBND subroutine code

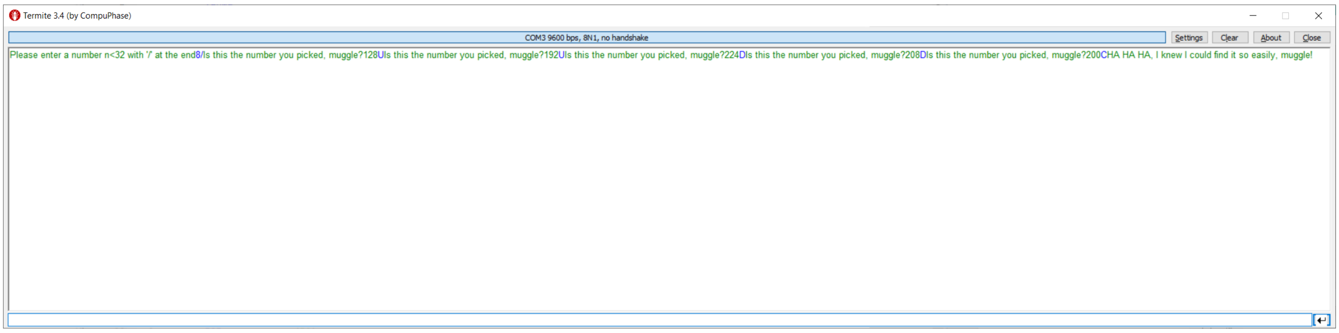
1.3.4 Main



```
;*****************************************************************
;LABEL       DIRECTIVE   VALUE       COMMENT
OFFSET      EQU         0x10
NUM     EQU         0x20000400
;*****************************************************************
; 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

;*****************************************************************
; Program section
;*****************************************************************
;LABEL       DIRECTIVE   VALUE       COMMENT
            AREA        main, READONLY, CODE
            THUMB
            EXTERN      OutStr  ; Reference external subroutine
            EXTERN      inputgetter
            EXTERN      upbnd
            EXPORT      __main  ; Make available

__main      PROC
start       LDR         R6,=NUM
            MOV         R7,#1   ;lower bound
            MOV         R8,#1   ;upper bound
            BL          inputgetter
            LSL         R8,R3   ;2^n calculation by shifting
            STR         R8,[R6]
            STR         R7,[R6,#4]
            BL          upbnd
finish      B           finish



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

Figure 1.6: Main code of the question 3.

1.3.5 Result



```
Please enter a number n<32 with '/' at the end8/Is this the number you picked, muggle?128UIs this the number you picked, muggle?192UIs this the number you picked, muggle?224DIs this the number you picked, muggle?208DIs this the number you picked, muggle?200CHA HA HA, I knew I could find it so easily, muggle!
```

Figure 1.7: Result of the question 3 seen in the terminal