**프로젝트에서 공통적으로 쓰는 코드 설명**

1. 여러 자리 수의 입력

* 여러 자리 수를 입력하기 위해 처음 입력된 숫자에다 10을 곱한 후, 두 번째 입력된 숫자를 더해주었다. 이렇게 만들어진 숫자에다 다시 10을 곱한 후, 세 번째 입력된 숫자를 더하는 것을 반복함으로써 여러 자리 수를 입력 받았다. 입력을 모두 마쳤을 경우에는 ENTER를 눌러서 종료한다.

<코드 설명>

mov r3, #10 *;r3는 ENTER 키가 입력되었는지를 판별해주는 REG*

INSERT\_NUM\_FIRST  *; 첫번째 자릿수 입력*

bl scan

sub r4, r0, #'0'

INSERT\_NUM *; 두 번째 자릿수 입력*

bl scan

CMP r0, r3 *; 만약 ENTER 키가 입력되었을 경우*

BEQ INSERT\_NUM2\_FIRST *; 입력 끝, 두 번째 변수를 입력 받으러 이동*

BNE MAKE\_DIGIT *; 숫자가 입력되었을 경우 \*10을 수행하는 branch로 이동*

MAKE\_DIGIT *; 입력된 숫자들에 자릿수를 부여해주는 branch*

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM

1. 나누기

* /2를 수행하는 경우 LSR을 사용하였다. Logical shift right by 1 bits는 값을 2로 나누는 것과 똑같기 때문이다. (코드 : mov r10, r10, LSR #1 )
* 다른 숫자로 나누기를 수행하는 경우, 나누는 수를 나누어지는 수에 몇 번 뺄 수 있는지를 count함으로써 몫과 나머지를 구한다.

<코드 설명>

*예시로 나누기 10을 하는 경우를 들었다. 다른 숫자로 나누고 싶을 경우에는 r3에다 원하는 숫자를 넣으면 된다*.

DIVIDE

sub r10, r10, r3 *;나누어지는 수에서 나누는 수를 뺀다*

add r2, r2, #1  *;몫을 1 증가시킨다*

CMP r10, r3 *; 나누어지는 수가 나누는 수보다 작을 경우에는*

BLT END\_DIVIDE *; 나눗셈을 끝낸다.*

bl DIVIDE *; 그렇지 않을 경우에는 계속 나눗셈을 진행한다.*

\* 나눗셈을 모두 진행하고 난 뒤의 나누어지는 수, 즉 r10의 값이 나눗셈의 나머지가 된다.

1. 여러 자리 수의 출력

* 여러 자리 수의 출력은 stmfd(REG -> MEM), ldmfd(MEM -> REG) 명령어를 이용하였다. Stmfd와 ldmfd는 메모리를 push하고 pop 할 때 방향이 반대라는 특징이 있다. 이를 활용해서 stmfd를 통해 일의 자리수부터 메모리에 각 값을 저장하였고, ldmfd를 통해 가장 차수가 높은 숫자부터 pop 시키고 print\_char을 반복하면서 여러 자리 수를 출력하였다. 이 때, 각 자리수의 숫자를 얻는 방법은 /10을 통해 가능하다. /10을 했을 때 나머지가 각 자릿수가 된다.

<코드 설명>

DIVIDE\_PRINT *; 나누기 10을 통해 몫과 나머지를 구함으로써 각 자릿수의 값을 얻는 과정(나누기에 대한 설명은 위에 자세히 있으므로 따로 하지 않겠다.)*

CMP r0, r3

BLT END\_DIVIDE\_PRINT

sub r0, r0, r3 ; r0 = i-10

add r2, r2, #1; share++

bl DIVIDE\_PRINT

END\_DIVIDE\_PRINT *; register의 값을 memory 에 저장하는 과정(push)*

stmfd sp!,{r0} ;r0의 값을 memory에 저장

add r7, r7, #1  *; stmfd가 몇번 수행되었는지 세는 COUNTER, 몇 번 ldmfd를 수행할 것인지 파악하기 위해 필요*

CMP r2, #0  *;몫이 0이 될 경우, 즉 모든 자릿수의 숫자를 다 구했을 경우*

BEQ PRINT\_NUM *; 출력을 담당하는 branch 로 이동*

mov r0, r2  *; 몫이 0이 아닐 경우 다시 나누기 10 수행*

mov r2, #0 *; initialize share*

bl DIVIDE\_PRINT

PRINT\_NUM *; 출력을 하는 branch*

ldmfd sp!,{r0} *;pop시켜서 최고차수의 자릿수 숫자부터 불러오기*

add r0, r0, #'0'  *;decimal to ascii*

bl print\_char  *;출력*

sub r7, r7, #1  *;COUNTER 1 감소*

CMP r7, #0  *;COUNTER=0이 될 경우 모든 자릿수 출력 완료-> 프로그램 종료.*

BEQ END\_START\_LOOP

BNE PRINT\_NUM  *;COUNTER !=0일 경우 다음 자릿수 출력하기 위해 PRINT\_NUM 재실행*

**프로젝트 1 -> 소수 개수 세기**

<알고리즘 설명> - C언어

텍스트이(가) 표시된 사진

자동 생성된 설명

<전체 코드 설명>

; ARM assenbler

AREA text, CODE

; This section is called "text", and contains code

ENTRY

IMPORT scan

IMPORT print

IMPORT print\_char

Start

*-----------------------------scan을 통해 첫 번째 소수 입력받기--------------------------------------*

mov r3, #10 ;r3 is enter

INSERT\_NUM\_FIRST

bl scan

sub r4, r0, #'0'

INSERT\_NUM

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM2\_FIRST

BNE MAKE\_DIGIT

MAKE\_DIGIT

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM

---------------------------*첫 번째 소수 입력 받기 끝 & 두 번째 소수 입력 받기*---------------------

INSERT\_NUM2\_FIRST

mov r5, r4 ; r5 is first prime number

bl scan

sub r6, r0, #'0'

INSERT\_NUM2

bl scan

CMP r0, r3 ;if enter is inserted

BEQ SET\_VARIABLE

BNE MAKE\_DIGIT2

MAKE\_DIGIT2

mov r2, #10

mul r6, r2, r6

sub r0, r0, #'0'

add r6, r6, r0

bl INSERT\_NUM2

-------------------*두 번째 소수 입력 받기 끝*----------------------------------------------------

SET\_VARIABLE

---------c언어에서 <for (int i = M; i <= N; i++) { count = 0; } > 부분 --------------------

mov r8, r6 ; r8 is second prime number

mov r9, r5 ;r5->r9, r9 is 'i', r5 is M

add r8, r8, #1 ;r8 -> r8+1, r8 was initially N

mov r10, #0 ;r10->0, r10 is count

mov r11, #1 ; r11 is 'j'

mov r1, #0 ;r1 is counter(몫)

mov r0, #0 ; r0 is count\_all

START\_LOOP

CMP r9, r8 ; i<=N

BEQ END\_START\_LOOP ; i>N일 경우 루프 밖을 벗어난다.

bl DIVIDE ; i<=N일경우 나눗셈 진행

HERE

CMP r10, #2 ;if(count==2)

BEQ UP\_COUNT\_ALL

add r9, r9, #1 ; i++

mov r5, r9;

mov r11, #1; initialize j

mov r10, #0 ; initialize count

bl START\_LOOP

UP\_COUNT\_ALL

add r0, r0, #1 ; count\_all ++

add r9, r9, #1 ; i++

mov r5, r9;

mov r11, #1; initialize j

mov r10, #0 ; initialize count

bl START\_LOOP

*-------------;for (int j = 1; j <= i; j++) {if (i%j == 0) {count++;}} 부분---------------------------------*

DIVIDE

CMP r11, r5 ; if j>i

BGT HERE

sub r9, r9, r11 ; r9 = i-j

add r1, r1, #1; counter++

CMP r9, r11 ; if i-j<j

BLT END\_DIVIDE

bl DIVIDE

END\_DIVIDE

CMP r9, #0

BEQ UP\_COUNT

bl NOTHING

UP\_COUNT ;remainder=0

add r10, r10, #1 ;count++

add r11, r11, #1 ; j++

mov r9, r5;

mov r1, #0;

bl DIVIDE

NOTHING ;remainder != 0

add r11, r11, #1 ; j++

mov r9, r5; initialize i

mov r1, #0; initialize counter

bl DIVIDE

*-------------------------소수의 개수인 r0를 출력하는 과정------------------------------*

END\_START\_LOOP

;we need to print r0, which is count\_all here

mov r2, #0 ; r2 is share

mov r7, #0 ;r7 is the number of saving memory

mov r3, #10 ; needed to divide

DIVIDE\_PRINT

CMP r0, r3 ; if i-10<10

BLT END\_DIVIDE\_PRINT

sub r0, r0, r3 ; r0 = i-10

add r2, r2, #1; share++

bl DIVIDE\_PRINT

END\_DIVIDE\_PRINT

stmfd sp!,{r0}

add r7, r7, #1

CMP r2, #0 ;if) share ==0

BEQ PRINT\_NUM

mov r0, r2 ;r2 -> r0 (r0 = share)

mov r2, #0 ; initialize share

bl DIVIDE\_PRINT

;DIVIDE\_PRINT\_END

PRINT\_NUM

ldmfd sp!,{r0}

add r0, r0, #'0' ;decimal to ascii

bl print\_char

sub r7, r7, #1

CMP r7, #0

BEQ finish

BNE PRINT\_NUM

*--------------------------------------소수 개수 출력 완료---------------------------------------*

finish

; Standard exit code: SWI 0x123456, calling routine 0x18

; with argument 0x20026

mov r0, #0x18

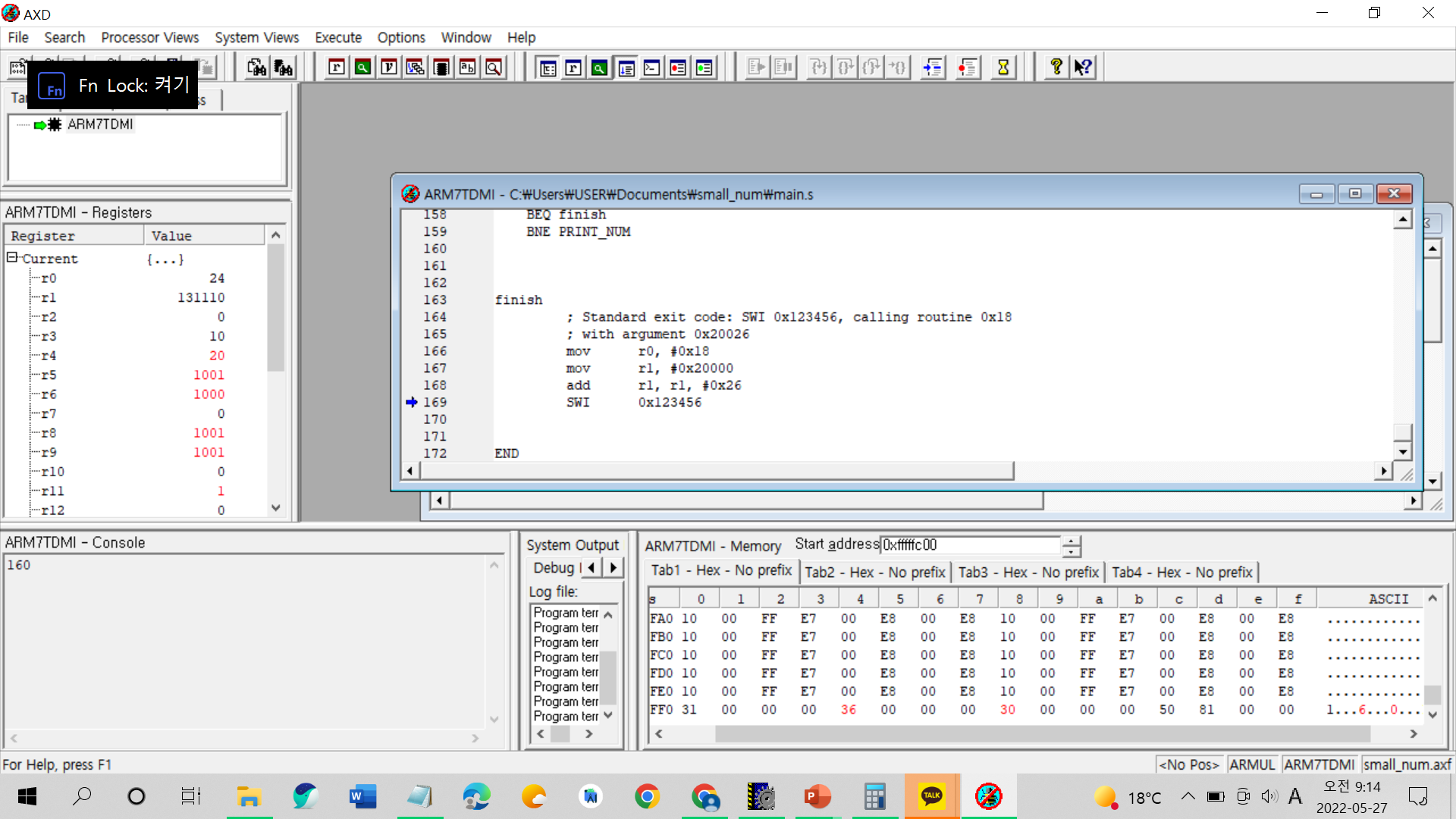
mov r1, #0x20000

add r1, r1, #0x26

SWI 0x123456

END

<구현 결과> -> 20과 1000사이의 소수 개수



**프로젝트 2 -> 이진 탐색 알고리즘**

<알고리즘 설명> - C언어

텍스트이(가) 표시된 사진

자동 생성된 설명

<주요 코드 설명>

10개의 숫자를 저장할 때 stack을 만들어서 memory 를 할당하였다.

* sub sp, sp, #40 ; allocate memory

<전체 코드 설명>

; ARM assenbler

AREA text, CODE

; This section is called "text", and contains code

ENTRY

IMPORT scan

IMPORT print

IMPORT print\_char

start

*----------------------숫자 10개 받는 과정(반복됨)-----------------------------------------------------*

;make stack & get 1st number

sub sp, sp, #40 ; allocate memory

mov r3, #10 ;r3 is enter

INSERT\_NUM\_FIRST

bl scan

sub r4, r0, #'0'

INSERT\_NUM

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM2\_FIRST

BNE MAKE\_DIGIT

MAKE\_DIGIT

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM

;get 2nd number

INSERT\_NUM2\_FIRST

mov r5, r4 ; r5 is first number

str r5, [sp, #0] ;store 1st number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM2

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM3\_FIRST

BNE MAKE\_DIGIT2

MAKE\_DIGIT2

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM2

;get 3rd number

INSERT\_NUM3\_FIRST

mov r5, r4 ; r5 is 2nd number

str r5, [sp, #4] ;store 2nd number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM3

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM4\_FIRST

BNE MAKE\_DIGIT3

MAKE\_DIGIT3

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM3

;get 4th number

INSERT\_NUM4\_FIRST

mov r5, r4 ; r5 is 3rd number

str r5, [sp, #8] ;store 3rd number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM4

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM5\_FIRST

BNE MAKE\_DIGIT4

MAKE\_DIGIT4

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM4

;get 5th number

INSERT\_NUM5\_FIRST

mov r5, r4 ; r5 is 4th number

str r5, [sp, #12] ;store 4th number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM5

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM6\_FIRST

BNE MAKE\_DIGIT5

MAKE\_DIGIT5

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM5

;get 6th number

INSERT\_NUM6\_FIRST

mov r5, r4 ; r5 is 5th number

str r5, [sp, #16] ;store 5th number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM6

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM7\_FIRST

BNE MAKE\_DIGIT6

MAKE\_DIGIT6

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM6

;get 7th number

INSERT\_NUM7\_FIRST

mov r5, r4 ; r5 is 6th number

str r5, [sp, #20] ;store 6th number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM7

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM8\_FIRST

BNE MAKE\_DIGIT7

MAKE\_DIGIT7

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM7

;get 8th number

INSERT\_NUM8\_FIRST

mov r5, r4 ; r5 is 7th number

str r5, [sp, #24] ;store 7th number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM8

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM9\_FIRST

BNE MAKE\_DIGIT8

MAKE\_DIGIT8

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM8

;get 9th number

INSERT\_NUM9\_FIRST

mov r5, r4 ; r5 is 8th number

str r5, [sp, #28] ;store 8th number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM9

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM10\_FIRST

BNE MAKE\_DIGIT9

MAKE\_DIGIT9

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM9

;get 10th number

INSERT\_NUM10\_FIRST

mov r5, r4 ; r5 is 9th number

str r5, [sp, #32] ;store 9th number in memory

bl scan

sub r4, r0, #'0'

INSERT\_NUM10

bl scan

CMP r0, r3 ;if enter is inserted

BEQ GET\_TARGET

BNE MAKE\_DIGIT10

MAKE\_DIGIT10

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM10

*------------------------숫자 10개 받기 완료--------------------------------------------------------*

*--------------------TARGET NUMBER 받기----------------------------------------------------*

GET\_TARGET

;get target number to find

mov r5, r4 ; r5 is 10th number

str r5, [sp, #36] ;store 10th number in memory

INSERT\_TARGET

bl scan

sub r4, r0, #'0'

INSERT\_TARGET\_NUM

bl scan

CMP r0, r3 ;if enter is inserted

BEQ SET\_VARIABLE

BNE MAKE\_DIGIT\_TARGET

MAKE\_DIGIT\_TARGET

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_TARGET\_NUM

-------------------------Target number 받기 완료---------------------------------------

SET\_VARIABLE

;set the variable

mov r5, r4 ; r5 is target number

mov r2, #0 ; r2 is low(index)

mov r3, #9 ; r3 is high(index)

mov r4, #0 ; r4 is mid (index)

;while

START\_LOOP

CMP r2, r3 ; low>high

BGT END\_START\_LOOP

add r6, r2, r3 ; r6 = high+low (index)

mov r6, r6, LSR #1 ; you need to make /2 here

mov r4, r6 ; mid= r4 = high+low /2 (index)

mov r8, #4

mul r7, r8, r4; index\*4

add r7, r7, sp;

ldr r8, [r7, #0] ;get the number that mid(r4) points

CMP r5, r8 ; if (target == data[mid])

BEQ SUCESS\_FINDING

CMP r5, r8 ; if (target < data[mid])

BLT CHANGE\_HIGH

CMP r5, r8 ; if (target > data[mid])

BGT CHANGE\_LOW

-------------------------------------탐색 성공-----------------------------------------

SUCESS\_FINDING

;we need to print r4(mid) here

mov r0, r8

mov r2, #0 ; r2 is share

mov r7, #0 ;r7 is the number of saving memory

mov r3, #10 ; needed to divide

---------------------------------탐색한 숫자 출력하는 과정---------------------------------------------

DIVIDE\_PRINT

CMP r0, r3 ; if i-10<10

BLT END\_DIVIDE\_PRINT

sub r0, r0, r3 ; r0 = i-10

add r2, r2, #1; share++

bl DIVIDE\_PRINT

END\_DIVIDE\_PRINT

stmfd sp!,{r0}

add r7, r7, #1

CMP r2, #0 ; if) share ==0

BEQ PRINT\_NUM

mov r0, r2 ;r2 -> r0 (r0 = share)

mov r2, #0 ; initialize share

bl DIVIDE\_PRINT

;DIVIDE\_PRINT\_END

PRINT\_NUM

ldmfd sp!,{r0}

add r0, r0, #'0' ;decimal to ascii

bl print\_char

sub r7, r7, #1

CMP r7, #0

BEQ END\_START\_LOOP

BNE PRINT\_NUM

-----------------------------출력 완료-------------------------------------------------------

CHANGE\_HIGH

sub r3, r4, #1 ; high = mid -1

bl START\_LOOP

CHANGE\_LOW

add r2, r4, #1 ; low = mid +1

bl START\_LOOP

END\_START\_LOOP

finish

; Standard exit code: SWI 0x123456, calling routine 0x18

; with argument 0x20026

mov r0, #0x18

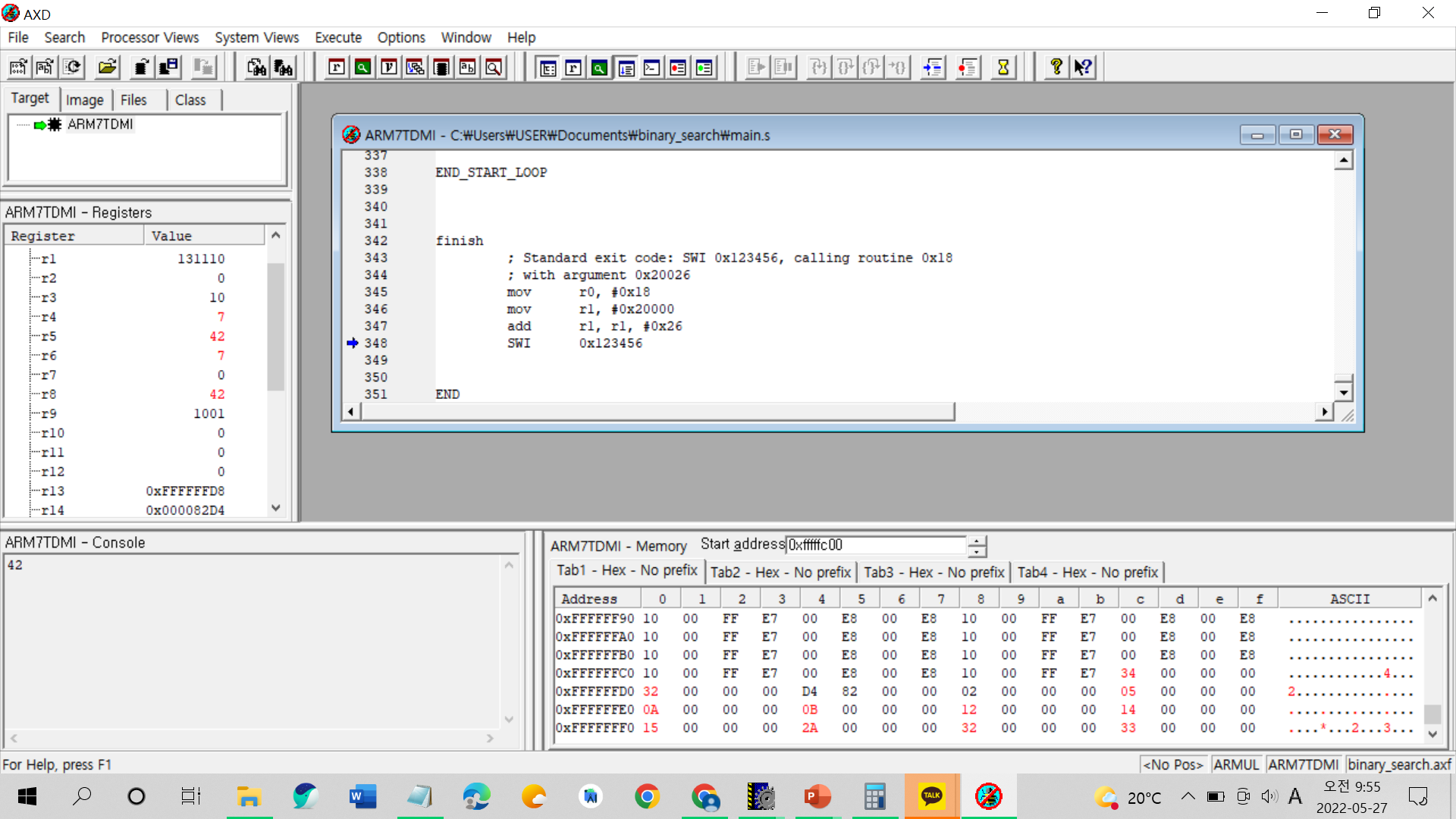
mov r1, #0x20000

add r1, r1, #0x26

SWI 0x123456

END

<구현 결과> - 2,5,10 , 11,18,20,21,42,50,51 에서 42를 찾는 과정



* **프로젝트 3 => Greedy**

<알고리즘 설명>

Greedy 알고리즘을 통해 최소의 수행횟수를 구하기 위해 B->A로 접근하는 방식을 택하였다. 만약 B가 짝수일 경우 /2를 수행하고 COUNT를 1 증가시킨다. 만약 B가 홀수이면서 일의 자리수가 1인 경우에는 1을 지우고 COUNT를 1 증가시킨다. 만약 B가 홀수이면서 다른 일의 자리 수를 가질 경우에는 (-1)을 출력한다. 뿐만 아니라 B->A로 접근하는 과정에서 B가 A보다 작아질 경우에도 (-1)을 출력한다. 조건에 맞는 명령문을 반복해서 수행하다가, A=B가 되면 COUNT+1을 출력한다.

<주요 코드 설명>

1. ODD, EVEN 판별

* 짝수의 경우는 2로 나누었을 때의 몫이 (짝수 +1)을 나누었을 때와의 몫과 같다. 홀수의 경우에는 2로 나누었을 때의 몫이 (홀수 +1)을 나누었을 때와의 몫과 다르다는 점을 이용해서 ODD, EVEN을 판별하였다. 나누기 2는 LSR로 쉽게 구현할 수 있으므로 이 방법을 고안하였다.

<코드 설명>

DECIDE\_EVEN\_ODD

add r11, r10, #1*; r11 = B+1 ; 짝, 홀을 판별하고 싶은 값에다가 1을 더한다.*

sub sp, sp, #4

str r10, [sp, #0*] ; 1을 더하지 않은 기존의 값을 추후에 쓸 예정이기 때문에 MEM에 저장*

mov r10, r10, LSR #1 *; r10 = B/2 기존값/2 를 진행한다*

mov r11, r11, LSR #1 *; r11 = (B+1)/2 (기존값+1)/2를 진행한다.*

CMP r10, r11

BEQ EVEN *; 몫이 같으면 짝수*

BNE ODD  *;몫이 다르면 홀수*

1. (-1) 출력 방법

* (-)을 출력하고, 1을 출력함으로써 (-1) 출력을 구현하였다.

<코드 설명>

mov r0, #45

bl print\_char  *; print (-)*

mov r0, #1;

add r0, r0, #'0' *; decimal to ascii*

bl print\_char  *; print (1)*

<전체 코드 설명>

; ARM assenbler

AREA text, CODE

; This section is called "text", and contains code

ENTRY

IMPORT scan

IMPORT print

IMPORT print\_char

start

-------------------------------A 입력 받기------------------------------------------

;get number A

mov r3, #10 ;r3 is enter

INSERT\_NUM\_FIRST

bl scan

sub r4, r0, #'0'

INSERT\_NUM

bl scan

CMP r0, r3 ;if enter is inserted

BEQ INSERT\_NUM2\_FIRST

BNE MAKE\_DIGIT

MAKE\_DIGIT

mov r2, #10

mul r4, r2, r4

sub r0, r0, #'0'

add r4, r4, r0

bl INSERT\_NUM

-------------------------------B 입력 받기------------------------------------------

;get number B

INSERT\_NUM2\_FIRST

mov r6, r4 ; r6 is number A

bl scan

sub r7, r0, #'0'

INSERT\_NUM2

bl scan

CMP r0, r3 ;if enter is inserted

BEQ SET\_VARIABLE

BNE MAKE\_DIGIT2

MAKE\_DIGIT2

mov r2, #10

mul r7, r2, r7

sub r0, r0, #'0'

add r7, r7, r0

bl INSERT\_NUM2

SET\_VARIABLE

mov r10, r7 ;r10 is number B

mov r1, #0 ; r1 is count

mov r2, #0 ; r2 is share(몫)

mov r3, #10 ; needed to divide

-------------------------------B 홀수, 짝수 판별------------------------------------------

START\_LOOP

CMP r6, r10 ; if(A==B)

BEQ END\_LOOP

BGT END\_START\_LOOP ;if(A>B)

BNE DECIDE\_EVEN\_ODD

DECIDE\_EVEN\_ODD

add r11, r10, #1; r11 = B+1

sub sp, sp, #4

str r10, [sp, #0] ; save r10 in stack

mov r10, r10, LSR #1 ; r10 = B/2

mov r11, r11, LSR #1 ; r11 = (B+1)/2

CMP r10, r11

BEQ EVEN

BNE ODD

EVEN

add r1, r1, #1 ;count++

bl START\_LOOP

ODD

ldr r10, [sp, #0] ; return r10(MEM) to r10(REG)

bl DIVIDE

-------------------------------B 홀수일 경우 나누기 진행------------------------------------------

;DIVIDE\_START

DIVIDE

sub r10, r10, r3 ; r10 = i-10

add r2, r2, #1; share++

CMP r10, r3 ; if i-10<10

BLT END\_DIVIDE

bl DIVIDE

-------------------------------B의 나머지가 1일 경우------------------------------------------

END\_DIVIDE

CMP r10, #1 ;if remainder==1

BEQ UP\_COUNT

BNE NOTHING

UP\_COUNT ;remainder==1

mov r10, r2 ;r2 -> r10 (r10 = share)

add r1, r1, #1 ; count++

bl START\_LOOP

-------------------------------B의 나머지가 1이 아닐 경우------------------------------------------

NOTHING ;else

;you need to print (-1) here

mov r0, #45

bl print\_char ; print (-)

mov r0, #1;

add r0, r0, #'0' ;decimal to ascii

bl print\_char ; print (1)

bl DONE

;DIVIDE\_END

-----------------count+1값 출력--------------------------------

END\_LOOP

;you need to print count here

add r1,r1, #1 ; count+1

mov r0, r1

add r0, r0, #'0'

bl print\_char

bl DONE

-------------------------A>B 인 상황일 경우----------------------------

END\_START\_LOOP

mov r0, #45

bl print\_char ; print (-)

mov r0, #1;

add r0, r0, #'0' ;decimal to ascii

bl print\_char ; print (1)

DONE

finish

; Standard exit code: SWI 0x123456, calling routine 0x18

; with argument 0x20026

mov r0, #0x18

mov r1, #0x20000

add r1, r1, #0x26

SWI 0x123456

END

<구현 결과> - 숫자 4, 42

텍스트, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

숫자 2, 162

텍스트, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

**util.s파일에 쓰인 함수들**

scan, print, print\_char 등의 함수는 util.s 파일에 저장해두어 각 프로젝트에서 IMPORT하여 사용하였다.