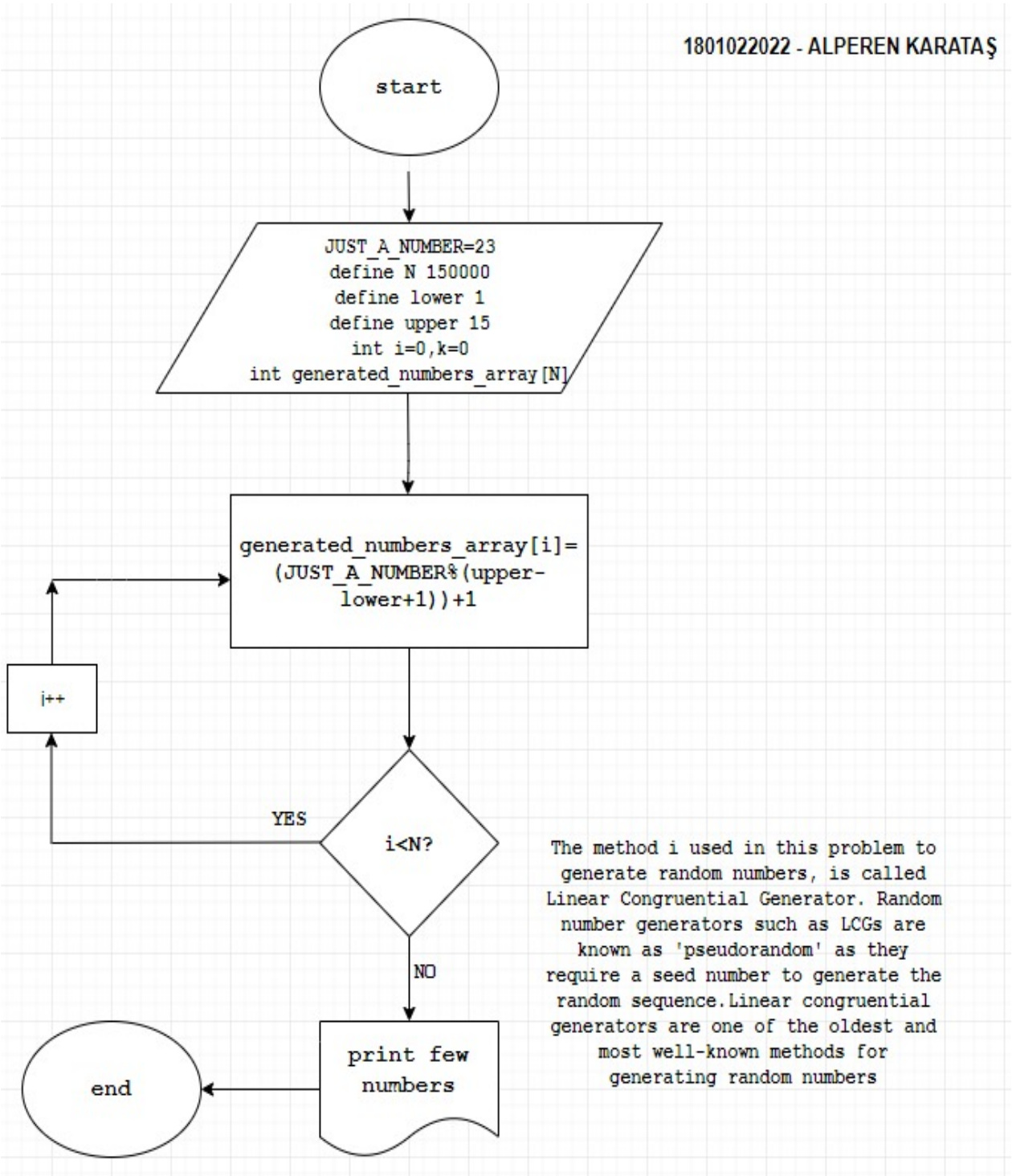
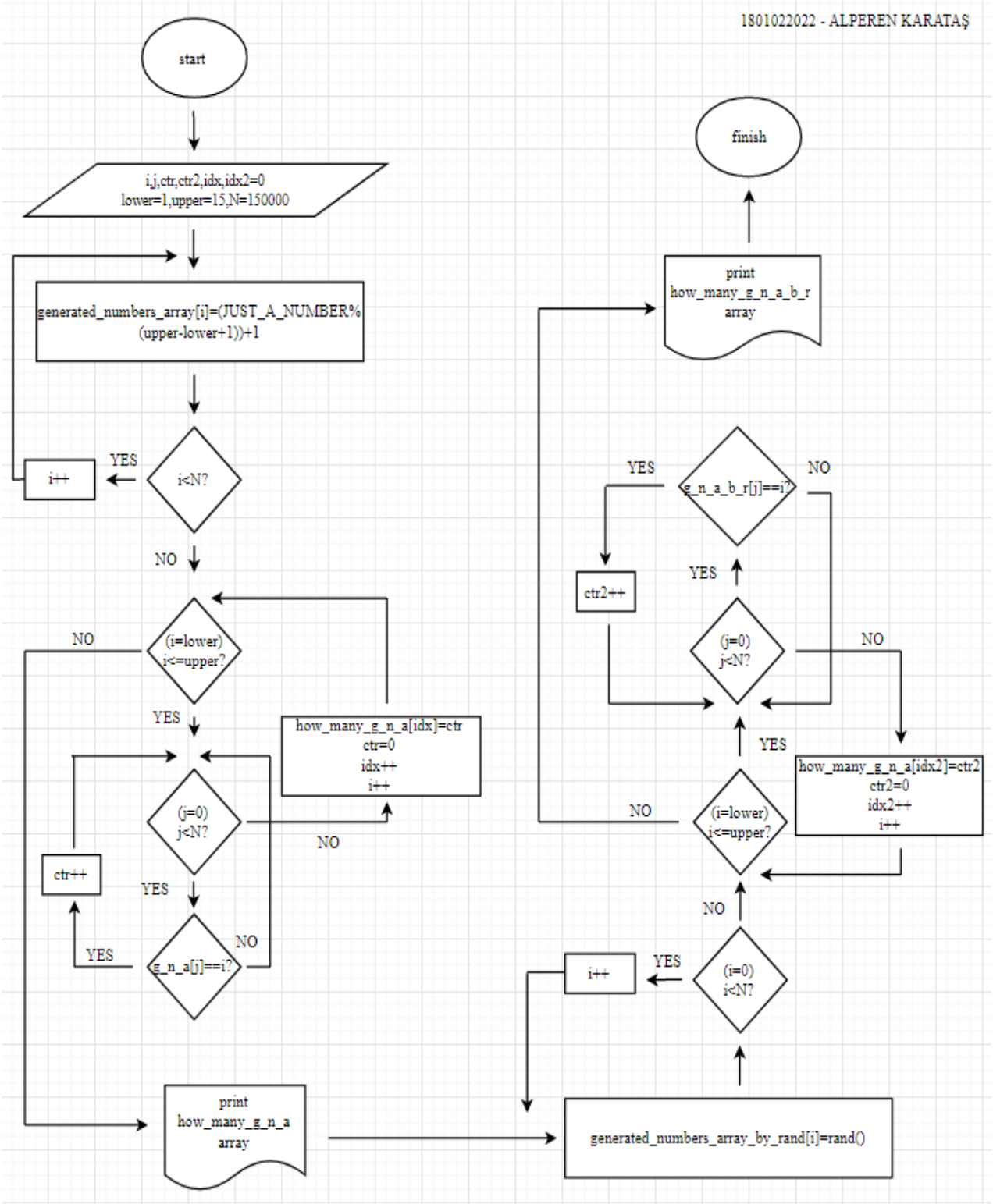
ELEC 334 HW#2

Problem 1. Pseudo-random number generator flowchart (I explained the method I used to generate random numbers in the lower right corner of the image below.)



Problem 2. Test your random number generator flowchart



Problem 3. Instruction Decode

NOTE: I explained the bit fields below the decoding.

ldr r5, [r6, #4]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	1	0	1	0	0	1	0	0	1	1	0	1	0	1

opA=bits[15:12]=0110

opB=bits[11:9]=1xx

imm5=bits[10:6]=00100 (decimal → 4)

Rn=R6=bits[5:3]=110 (decimal → 6)

Rt=R5=bits[5:2]=101 (decimal → 6)

Hexadecimal representation: 0x6935

mvns r4, r4

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	0	1	1	1	1	1	0	0	1	0	0

bits[15:10]=010000

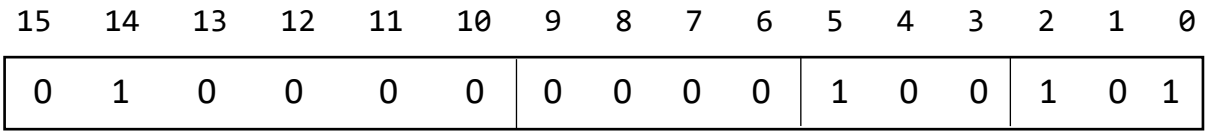
opcode=bits[9:6]=1111

Rm=R4=bits[5:3]=100 (decimal → 4)

Rd=R4=bits[2:0]=100 (decimal → 4)

Hexadecimal representation: 0x43E4

ands r5, r5, r4 (ands r5, r4)



bits[15:10]=010000

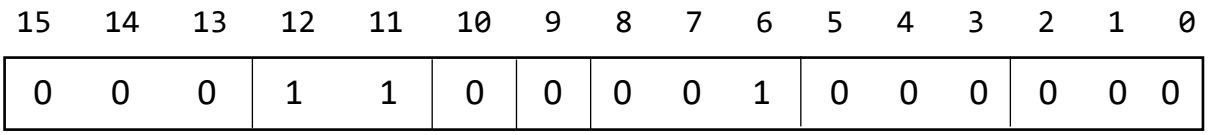
opcode=bits[9:6]=0000

Rm=R4=bits[5:3]=100 (decimal → 4)

Rdn=R5=bits[2:0]=101 (decimal → 5)

Hexadecimal representation: 0x4025

adds r0, r0, r1



bits[15:14]= 00

opcode=bits[13:9]=01100

Rm=R1=bits[8:6]=001 (decimal → 1)

Rn=R0=bits[5:3]=000 (decimal → 0)

Rd=R0=bits[2:0]=000 (decimal → 0)

Hexadecimal representation: 0x1840

add r0, r0, r1 (ignored)

subs r2, r4, #2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	1	1	0	1	0	1	0	0	0	1	0

bits[15:14]= 00

opcode=bits[13:9]=01101

imm3=bits[8:6]=010 (decimal → 2)

Rn=R4=bits[5:3]=100 (decimal → 4)

Rd=R2=bits[2:0]=010 (decimal → 1)

Hexadecimal representation: 0x1EA2

asrs r2, r4, #21

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	0	1	0	1	0	1	1	0	0	0	1	0

bits[15:14]= 00

opcode=bits[13:9]=010xx

imm5=bits[10:6]=10101 (decimal → 21)

Rm=R4=bits[5:3]=100 (decimal → 4)

Rd=R2=bits[2:0]=010 (decimal → 2)

Hexadecimal representation: 0x1562

str r5, [r6, r1]

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	1	0	0	0	0	0	1	1	1	0	1	0	1

opA=bits[15:12]=0101

opB=bits[11:9]=000

Rm=R1=bits[8:6]=001 (decimal → 1)

Rn=R6=bits[5:3]=110 (decimal → 6)

Rt=R5=bits[2:0]=101 (decimal → 5)

Hexadecimal representation: 0x5075

bx lr

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	1	0	0	0	1	1	1	0	1	1	1	0	(0)	(0)	(0)

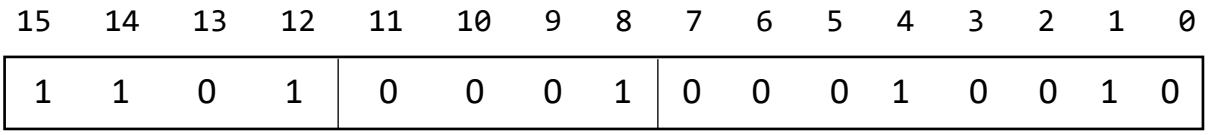
bits[15:10]= 010001

opcode=bits[9:6]=110x

Rm=R14=bits[6:3]=1110 (decimal → 14) (lr:Link register → R14)

Hexadecimal representation: 0x4770

bne 0x12



bits[15:12]=1101

condition(NE mnemonic)=bits[11:8]=0001

imm8=bits[7:0]=00010010 (hex → 0x12)

Hexadecimal representation: 0xD112

Problem 4. Instruction Cycle Times

*****From the ARM-Appendix B3 TABLE B3.14 ARM11 (ARMv6) instruction cycle timings model/Instruction-set-summary and**

https://developer.arm.com/documentation/ddi0484/b/Programmers- *******

ldr r5, [r6, #4] → cycles : 2

(instruction class: LDR *pc*, [*sp*, # *off*])

mvns r4, r4 → cycles : 1

(instruction class: ALU operations except a MOV to *pc* (for MOV to *pc*, see BX)

ands r5, r5, r4 (ands r5, r4) → cycles : 1

(instruction class: ALU operations except a MOV to *pc* (for MOV to *pc*, see BX)

adds r0, r0, r1 → cycles : 1

(instruction class: ALU operations except a MOV to *pc* (for MOV to *pc*, see BX)

add r0, r0, r1 (ignored)

subs r2, r4, #2 → cycles : 1

(instruction class: ALU operations except a MOV to *pc* (for MOV to *pc*, see BX)
asrs r2, r4, #21 → cycles : 1

(instruction class: ALU operations except a MOV to *pc* (for MOV to *pc*, see BX)

str r5, [r6, r1] → cycles : A → 2

(instruction class: STR → [Rd],[*Rn, Rm*])

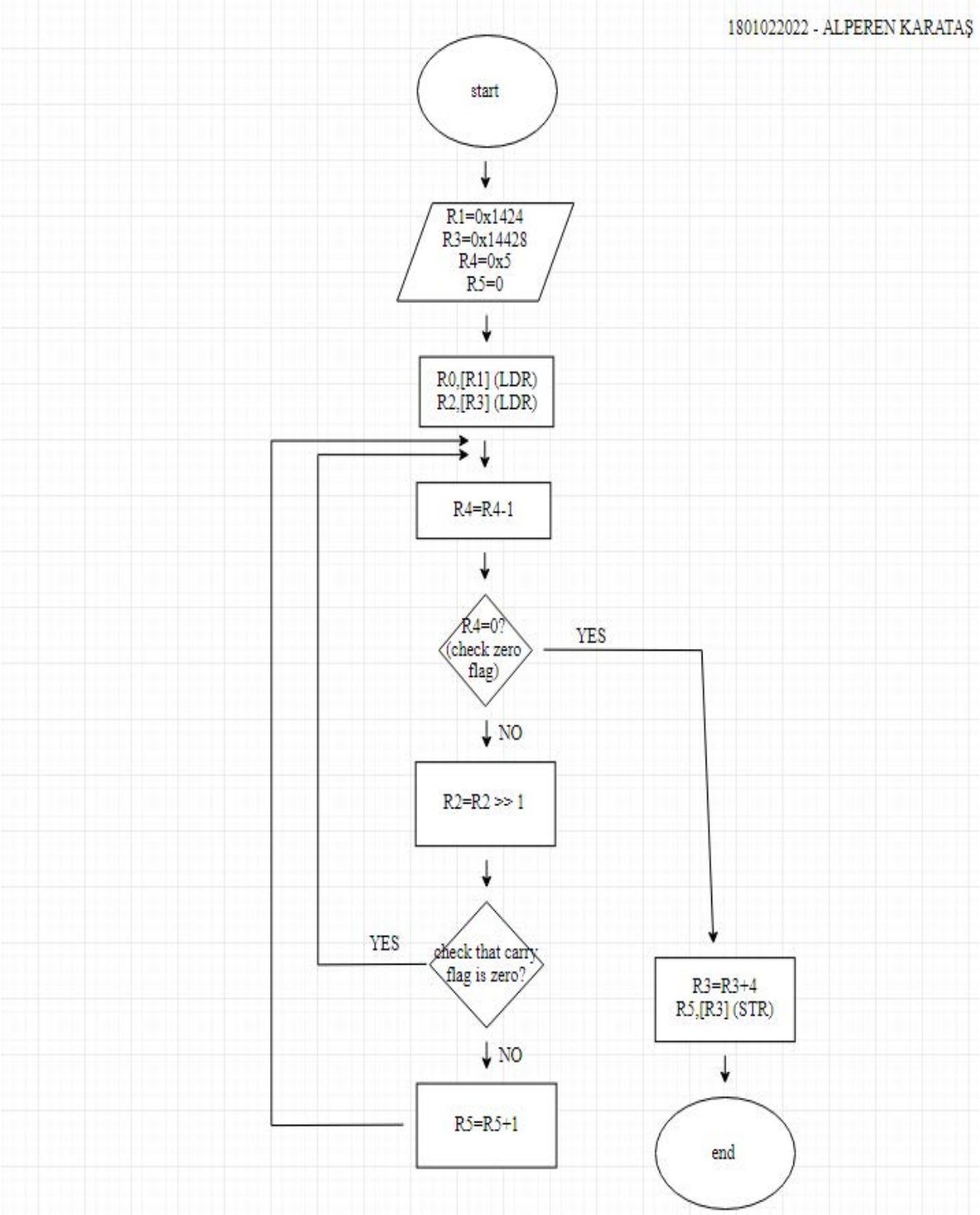
bx lr → cycles : 2

(instruction class : BX *lr)*

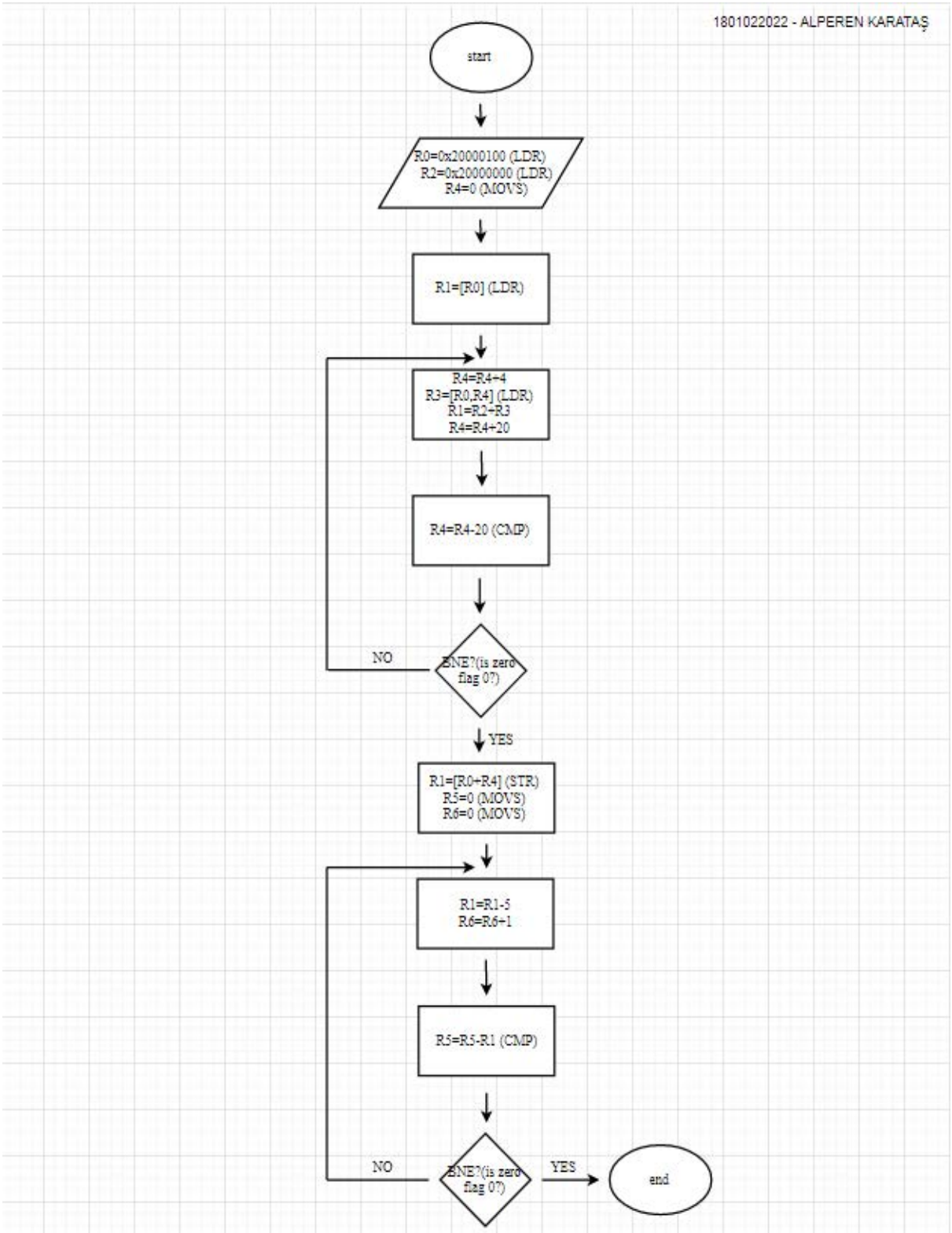
bne 0x12 → cycles : 1

(instruction class: B <cc> <*immed*>*)*

Problem 7. Assembly Hamming distance flowchart



Problem 8. Assembly Average flowchart



!!!Codes start from the next page!!!

Problem 1.

```
/* main.c
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 1
* Pseudo-random number generator
*
*/

#include <stdio.h>

#include <stdlib.h>

#include "myrand.h"

#define N 150000       //The number of total generated numbers

#define lower 1        //My lower limit for random number generation

#define upper 15       //My upper limit for random number generation


int generated_numbers_array[N];


int main()

{

    int i=0,k=0;


    while(i<N){

        generated_numbers_array[i]=myrand(lower,upper);     //Numbers
generated in myrand function in main.c are thrown into the array.

        i++;

    }


    printf("Problem 1:\n\nA few numbers from myrand():\n");

    for(int a=0;a<3;a++){                                   //Printing
the first three numbers of array. Just for verification.

        printf("myrand\n%d\n",generated_numbers_array[a]);

    }

}
```

```c
/* myrand.c
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 1
* Pseudo-random number generator
*
*/

#include <stdio.h>

#include <stdlib.h>

#include "myrand.h"


/*The methods name is Linear Congruential Generator, which i explained
in flowchart diagram in the pdf.*/


int JUST_A_NUMBER=23;  // No specific reason associated with the code,
it's my hometowns plate number.


int myrand(int lower,int upper){


    JUST_A_NUMBER = ((JUST_A_NUMBER * 8428) + 1) % 78956; // Linear
Congruential Generator

    return (JUST_A_NUMBER %(upper-lower+1))+lower;


}
```

```
/* myrand.h
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 1
* Pseudo-random number generator
*
*/

#include <stdio.h>

#include <stdlib.h>

#include "myrand.h"


#ifndef MYRAND_H_INCLUDED

#define MYRAND_H_INCLUDED


int myrand(int,int);


#endif // MYRAND_H_INCLUDED
```

Problem 2.

```
/* main.c
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 2
* Testing random number generator
*
*/

#include <stdio.h>

#include <stdlib.h>

#include "myrand.h"

#include "test_random.h"

#define N 150000            //The number of total generated numbers

#define lower 1             //My lower limit for random number
generation

#define upper 15            //My upper limit for random number
generation

#define start 1             //Used in printing

#define stop 15             //Used in printing


/*Arrays which i use for keep the values and print their rates.*/

int generated_numbers_array[N];

int how_many_of_each_number[N];

int generated_numbers_array_by_rand[N];     //for rand() func.

int how_many_of_each_number_by_rand[N];     //for rand() func.


int main()

{

    int i=0,k=0;

    int ctr,ctr2=0;

    int idx,idx2=0;

    int m,n;

    int x,y;
```

```
/*Here is for myrand() function*/


    while(i<N){
        generated_numbers_array[i]=myrand(lower,upper);  //Numbers
generated in myrand function in main.c are thrown into the array.
        i++;
    }


    for(int i=lower;i<=upper;i++){                       //Calculating
how many numbers are produced between the specified limits.
        for(int j=0;j<N;j++){
            if(generated_numbers_array[j]==i)
                ctr++;
        }
        how_many_of_each_number[idx]=ctr;              //Calculated
values are kept in array one by one.
        idx++;
        ctr=0;


    }


    printf("Problem 2:\n\nresults from myrand():\n\n");     //The
calculated values are printed as desired.(Their ratios are calculated.)
    for(m=start,n=0; m<stop,n<stop; m++,n++)
        printf("%d: %.3f    ",m,(float)how_many_of_each_number[n]/N*10);


/*Here is for myrand() function. Almost the same, not much different*/


    while(k<N){
        generated_numbers_array_by_rand[k]=rand_func(lower,upper);
//Numbers generated in rand function in main.c are thrown into the
array.
        k++;
    }
```

```
for(int i=lower;i<=upper;i++){                          //Numbers
generated in myrand function in main.c are thrown into the array.
        for(int j=0;j<N;j++){
            if(generated_numbers_array_by_rand[j]==i)
                ctr2++;
        }
        how_many_of_each_number_by_rand[idx2]=ctr2;         //Calculated
values are kept in array one by one.
        idx2++;
        ctr2=0;
    }


    printf("\n\nresults from rand():\n\n");                 //The
calculated values are printed as desired.(Their ratios are calculated.)
    for(x=start,y=0; x<stop,y<stop; x++,y++)
        printf("%d: %.3f
",x,(float)how_many_of_each_number_by_rand[y]/N*10);


    printf("\n");


    return 0;
}
```

```c
/* myrand.c
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 2
* Testing random number generator
*
*/

#include <stdio.h>

#include <stdlib.h>

#include "myrand.h"


/*The methods name is Linear Congruential Generator, which i explained
in flowchart diagram in the pdf.*/


int JUST_A_NUMBER=23;   // No specific reason associated with the code,
it's my hometowns plate number.


int myrand(int lower,int upper){


    JUST_A_NUMBER = ((JUST_A_NUMBER * 8428) + 1) % 78956;    // Linear
Congruential Generator

    return (JUST_A_NUMBER %(upper-lower+1))+lower;


}
```

```
/* myrand.h
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 2
* Testing random number generator
*
*/

#ifndef MYRAND_H_INCLUDED

#define MYRAND_H_INCLUDED


int myrand(int,int);


#endif // MYRAND_H_INCLUDED
```

```c
/* test_random.c
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 2
* Testing random number generator
*
*/

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include "test_random.h"


int rand_func(int lower,int upper){


    return (rand() %(upper-lower+1))+lower;   //Simple rand function, as
we already know.


}
```

```
/* test_random.h
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 2
* Testing random number generator
*
*/

#ifndef TEST_RANDOM_H_INCLUDED

#define TEST_RANDOM_H_INCLUDED


int rand_func(int,int);


#endif // TEST_RANDOM_H_INCLUDED
```

Problem 5.

```
/* p5.s
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 5
* Assembly delay function
*
*/

    ; Edit below this line

      MOVS R0,#8

loop

      SUBS R0,#1

      BNE loop

    ; Edit above this line

    B .

    ENDP


    END
```

Problem 6.

```
/* p6.s
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 6
* Assembly LED toggle
*
*/

GPIOB_BASE EQU       0x50000400

GPIOB_ODR EQU       GPIOB_BASE + 0x14


LDR R6, =GPIOB_ODR

      LDR R5, [R6]

blink

      LDR R4,=0x0000       ;led toggles,remember active low

      ANDS R5, R5, R4

      STR R5, [R6]         ;in this situation light is on

      LDR R0, =0x7A1200

led_on                           ;delays 1 second with the light

      SUBS R0,R0,#1

      BNE led_on

      LDR R4,=0x1000       ;led toggles,remember active low

      ORRS R5, R5, R4

      STR R5, [R6]         ;in this situation light went out

      LDR R0, =0x7A1200

led_off                          ;delays 1 second without light

      SUBS R0,R0,#1

      BNE led_off

      B blink

    ; Edit above this line

    B .

    ENDP

    END
```

Problem 7.

```
/* p7.s
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 7
* Assembly Hamming distance
*
*/

    ; Edit below this line

    LDR R1,=0x14224

    LDR R0,[R1]

    LDR R3,=0x14228

    LDR R2,[R3]

    EORS R2,R2,R0

    ;Assume that the numbers are 5 bits

    MOVS R4,#0x5

    MOVS R5,#0 ;counter of 1's.

loop

      SUBS R4,R4,#1

      CMP R4,#0

      BNE finish   ;if zero flag is clear,branch to finish

      LSRS R2,R2,#1

      BCC loop      ;if carry flag is clear,branch to loop

      ADDS R5,R5,#1

      B loop

finish

      ADDS R3,R3,#4

      STR R5,[R3]   ;write the result to the R3 memory address

    ; Edit above this line

    B .

    ENDP

    END
```

Problem 8.

```
/* p8.s
*
* Prepared by:
* Alperen Karataş
*
* Notes:
* ELEC334 2020 Fall HW2 - Problem 8
* Assembly Average
*
*/

    ; Edit below this line

    LDR R0,=0x20000100

    LDR R2,=0x20000000

    MOVS R4,#0

      LDR R1,[R0]

loop

    ADDS R4,#4

    LDR R3,[R0,R4]

    ADDS R1,R1,R3

    CMP R4,#20

    BNE loop

    STR R1,[R0,R4]

    MOVS R5,#0

    MOVS R6,#0

divide

    SUBS R1,R1,#5

    ADDS R6,R6,#1

    CMP R5,R1

    BNE divide

    STR R6,[R2]

    ; Edit above this line

    B .

    ENDP

    END
```

Kaynaklar

1. https://diagrams.freebusinessapps.net/flow-chart
2. https://developer.arm.com/documentation/ddi0484/b/Programmers-Model/Instruction-set-summary
3. https://stackoverflow.com/
4. RM0444 Reference Manual
5. ARMv6-M Architecture Reference Manual