**תרגיל בית 3**

שאלה 1

שפת C:

Int sum = 0;

For(int i = R1,i <= R1 + R2 - 1 ,i++)

{

    if(Ram[i] % 2 == 0)

        sum = sum + Ram[i];

}

R0 = sum;

שפת psodocode:

int sum = 0;

int i = R1;

WHILE:

    if(i > R1 + R2 - 1) goto STOP;

        if(Ram[i] & 1 != 0) goto ODD

            sum = sum + Ram[i];

        ODD:

            i = i + 1;

        goto WHILE;

    STOP:

R0 = sum;

END:

שפתAssembly :

//int sum = 0;

@sum

M = 0;

//int i = R1;

@R1

D=M

@i

M=D

//WHILE:

(WHILE)

//if(i > R1 + R2) goto STOP;

@R1

D=M - 1 //D = R1 - 1

@R2

D = D + M //R1+R2 - 1

@i

D = M - D //i - (R1-R2-1)

@STOP

D;JGT//goto STOP

//if(Ram[i] & 1 != 0) goto ODD

@1

D = A //D=1

@i

A = M //Ram[i]

D = M & D //Ram[i] & 1

@ODD

D;JNE//if D != 0 jump to ODD

//sum = sum + Ram[i];

@i

A = M

D = M //D = Ram[i]

@sum

M = M + D //sum = sum + Ram[i];

//ODD:

(ODD)

//i = i + 1;

@i

M = M + 1

//goto WHILE;

@WHILE

0;JMP

//STOP:

(STOP)

//R0 = sum;

@sum

D = M

@R0

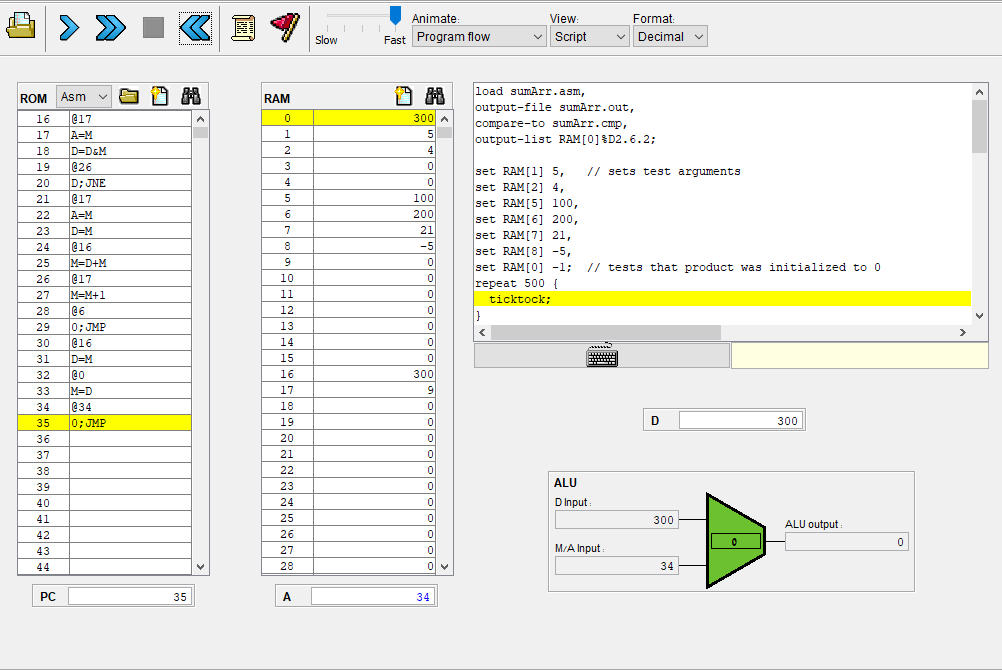
M = D

//END:

(END)

@END

0;JMP



שאלה 2

(א)

שפת C:

int multself(int i);

{

    int mult = 0;

    for(int j = 0; j < i; j++)

    {

        mult = mult + i;

    }

    return mult;

}

שפת psodocode:

MULTSELF:

mult = 0;

i2=\*(SP+1);

j = 0

WHILE:

    if(j >= i2) goto MULTEND;

        mult = mult + i2;

    j++;

goto WHILE;

MULTEND:

push(mult);

goto RET1;

שפתAssembly :

//MULTSELF:

(MULTSELF)

//mult = 0;

@mult

M = 0

//i2=\*(SP+1);

@1

D = A

@SP

A = D + M

D = M

@i2

M = D

//j = 0;

@j

M = 0

//WHILE:

(WHILE)

//if(j >= i) goto MULTEND;

@i2

D = M

@j

D = M - D // j - i2 >= 0

@MULTEND

D;JGE

//mult = mult + i2;

@i2

D = M

@mult

M = M + D

//j++;

@j

M = M + 1

//goto WHILE;

@WHILE

0;JMP

//MULTEND:

(MULTEND)

//push(mult);

@mult

D = M

@SP

M = M - 1

A = M

M = D

//goto RET2;

@SP

A = M + 1

A = M

0;JMP

(ב)

שפת C:

int count\_square (int numArr[], int size)

{

    int index = numArr;

    int size\_f = size;

    int count = 0;

    for(int i3 = index ; i3 < index + size ; i3++)

    {

        num = Ram[i3];

        for(int j2 = 0; multself(j2) < num ; j2++)

        {

            if(multself(j2) == num)

                count++;

        }

    }

    return count;

}

שפת psodocode:

COUNT\_SQUARE:

count = 0;

j2 = 0;

index = &numArr;

i3 = index;

size\_f = size;

WHILE1:

    if(i3 >= index + size\_f) goto WHILE1\_END

    num = Ram[i3];

    j2 = 0;

    WHILE2:

        push(j2);

        push(&RET2);

        goto MULTSELF;

        RET2:

        i\_mult\_i = pop();

        SP += 2

        if(i\_mult\_i > num) goto WHILE2\_END

            if(i\_mult\_i != num) goto SKIP\_COUNT

                count++;

        SKIP\_COUNT:

        j2++;

        goto WHILE2;

        WHILE2\_END:

        i3++;

        goto WHILE1;

    WHILE1\_END:

    push(count);

    goto RET1;

שפתAssembly :

//COUNT\_SQUARE:

(COUNT\_SQUARE)

//count = 0;

@count

M = 0

//j2 = 0;

@j2

M = 0

//index=\*(SP+1);

@1

D = A

@SP

A = D + M

D = M

@index

M = D

//i3 = index

@index

D=M

@i3

M=D

//size\_f=\*(SP+2);

@2

D = A

@SP

A = D + M

D = M

@size\_f

M = D

//WHILE1:

(WHILE1)

//if(i3 >= index + size\_f) goto WHILE1\_END

@size\_f

D = M

@index

D = D + M //index + size\_f

@i3

D = M - D

@WHILE1\_END

D;JGE

//num = Ram[i3];

@i3

A = M //Ram[i3]

D = M

@num

M = D

//j2 = 0;

@j2

M = 0

//WHILE2:

(WHILE2)

//push(j2);

@j2

D = M

@SP

M = M-1

A = M

M = D

//push(&RET2);

@RET2

D = A

@SP

M = M - 1

A = M

M = D

//goto MULTSELF;

@MULTSELF

0;JMP

//RET2:

(RET2)

//i\_mult\_i = pop();

@SP

A = M

D = M // D = \*SP

@SP

M = M + 1

@i\_mult\_i

M = D

//SP += 2;

@2

D=A

@SP

M = D + M

//if(i\_mult\_i >= num) goto WHILE2\_END

@num

D = M

@i\_mult\_i

D = M - D // i\_mult\_i - num > 0

@WHILE2\_END

D;JGT

//if(i\_mult\_i != num) goto SKIP\_COUNT

@num

D = M

@i\_mult\_i

D = M - D // i\_mult\_i - num != 0

@SKIP\_COUNT

D;JNE

//count++;

@count

M = M + 1

//SKIP\_COUNT:

(SKIP\_COUNT)

//j2++;

@j2

M = M + 1

//goto WHILE2;

@WHILE2

0;JMP

//WHILE2\_END:

(WHILE2\_END)

//i3++;

@i3

M = M + 1

//goto WHILE1;

@WHILE1

0;JMP

//WHILE1\_END:

(WHILE1\_END)

//push(count);

@count

D = M

@SP

M = M - 1

A = M

M = D

//goto RET1;

@SP

A = M + 1

A = M

0;JMP

(ג)

שפת C:

int main()

{

    int \*SP = &SCREEN;

    count\_square (int numArr[], int size);

    return 0;

}

int count\_square (int numArr[], int size)

{

    int index = numArr;

    int size\_f = size;

    int count = 0;

    for(int i3 = index ; i3 < index + size ; i3++)

    {

        num = Ram[i3];

        for(int j2 = 0; multself(j2) < num ; j2++)

        {

            if(multself(j2) == num)

                count++;

        }

    }

    return count;

}

int multself(int i);

{

    int mult = 0;

    for(int j = 0; j < i; j++)

    {

        mult = mult + i;

    }

    return mult;

}

שפת psodocode:

SP = &SCREEN;

push(R2);

push(R1);

push(&RET1);

goto COUNT\_SQUARE;

RET1:

R3 = pop();

SP += 3

goto END;

COUNT\_SQUARE:

count = 0;

j2 = 0;

index = &numArr;

i3 = index;

size\_f = size;

WHILE1:

    if(i3 >= index + size\_f) goto WHILE1\_END

    num = Ram[i3];

    j2 = 0;

    WHILE2:

        push(j2);

        push(&RET2);

        goto MULTSELF;

        RET2:

        i\_mult\_i = pop();

        SP += 2

        if(i\_mult\_i > num) goto WHILE2\_END

            if(i\_mult\_i != num) goto SKIP\_COUNT

                count++;

        SKIP\_COUNT:

        j2++;

        goto WHILE2;

        WHILE2\_END:

        i3++;

        goto WHILE1;

    WHILE1\_END:

    push(count);

    goto RET1;

MULTSELF:

mult = 0;

i2=\*(SP+1);

j = 0

WHILE:

    if(j >= i2) goto MULTEND;

        mult = mult + i2;

    j++;

goto WHILE;

MULTEND:

push(mult);

goto RET2;

END:

שפתAssembly :

//SP = &SCREEN;

@SCREEN

D = A

@SP

M = D

//push(R2);

@R2

D = M

@SP

M = M-1

A = M

M = D

//push(R1);

@R1

D = M

@SP

M = M-1

A = M

M = D

//push(&RET1);

@RET1

D = A

@SP

M = M - 1

A = M

M = D

//goto COUNT\_SQUARE;

@COUNT\_SQUARE

0;JMP

//RET1:

(RET1)

//R3 = pop();

@SP

A = M

D = M // D = \*SP

@SP

M = M + 1

@R3

M = D

//SP += 3;

@3

D=A

@SP

M = D + M

//goto END;

@END

0;JMP

//COUNT\_SQUARE:

(COUNT\_SQUARE)

//count = 0;

@count

M = 0

//j2 = 0;

@j2

M = 0

//index=\*(SP+1);

@1

D = A

@SP

A = D + M

D = M

@index

M = D

//i3 = index

@index

D=M

@i3

M=D

//size\_f=\*(SP+2);

@2

D = A

@SP

A = D + M

D = M

@size\_f

M = D

//WHILE1:

(WHILE1)

//if(i3 >= index + size\_f) goto WHILE1\_END

@size\_f

D = M

@index

D = D + M //index + size\_f

@i3

D = M - D

@WHILE1\_END

D;JGE

//num = Ram[i3];

@i3

A = M //Ram[i3]

D = M

@num

M = D

//j2 = 0;

@j2

M = 0

//WHILE2:

(WHILE2)

//push(j2);

@j2

D = M

@SP

M = M-1

A = M

M = D

//push(&RET2);

@RET2

D = A

@SP

M = M - 1

A = M

M = D

//goto MULTSELF;

@MULTSELF

0;JMP

//RET2:

(RET2)

//i\_mult\_i = pop();

@SP

A = M

D = M // D = \*SP

@SP

M = M + 1

@i\_mult\_i

M = D

//SP += 2;

@2

D=A

@SP

M = D + M

//if(i\_mult\_i >= num) goto WHILE2\_END

@num

D = M

@i\_mult\_i

D = M - D // i\_mult\_i - num > 0

@WHILE2\_END

D;JGT

//if(i\_mult\_i != num) goto SKIP\_COUNT

@num

D = M

@i\_mult\_i

D = M - D // i\_mult\_i - num != 0

@SKIP\_COUNT

D;JNE

//count++;

@count

M = M + 1

//SKIP\_COUNT:

(SKIP\_COUNT)

//j2++;

@j2

M = M + 1

//goto WHILE2;

@WHILE2

0;JMP

//WHILE2\_END:

(WHILE2\_END)

//i3++;

@i3

M = M + 1

//goto WHILE1;

@WHILE1

0;JMP

//WHILE1\_END:

(WHILE1\_END)

//push(count);

@count

D = M

@SP

M = M - 1

A = M

M = D

//goto RET1;

@SP

A = M + 1

A = M

0;JMP

//MULTSELF:

(MULTSELF)

//mult = 0;

@mult

M = 0

//i2=\*(SP+1);

@1

D = A

@SP

A = D + M

D = M

@i2

M = D

//j = 0;

@j

M = 0

//WHILE:

(WHILE)

//if(j >= i) goto MULTEND;

@i2

D = M

@j

D = M - D // j - i2 >= 0

@MULTEND

D;JGE

//mult = mult + i2;

@i2

D = M

@mult

M = M + D

//j++;

@j

M = M + 1

//goto WHILE;

@WHILE

0;JMP

//MULTEND:

(MULTEND)

//push(mult);

@mult

D = M

@SP

M = M - 1

A = M

M = D

//goto RET2;

@SP

A = M + 1

A = M

0;JMP

//END:

(END)

@END

0;JMP

