**An**

**Assignment**

**On**

**“UVA Problem Solutions Using C and Assembly Language”**

**Assembly Language Laboratory**

**[ CSE 2208]**

**Submitted To: -**

Mr. Sarfaraz Newaz

Lecturer

Computer Science and Engineering Discipline,

Khulna University.

**Submitted By: -**

Md Rakibul Hasan Molla

Student ID: 1602006

Year: 3rd Term: II

Computer Science and Engineering Discipline,

Khulna University.

Date: 25 October 2018



Computer Science and Engineering Discipline,

Khulna University

**Index**

1. Hashmat the Brave Warrior -10055
2. Back to High School Physics -10071
3. Pizza Cutting -10079
4. The Land of Justice - 10499
5. Big Chocolate-10970
6. Relational Operator -11172
7. Horror Dash-11799
8. Bafana Bafana-11805
9. Peter’s Smokes-10346
10. F91-10696
11. Odd sum -10783
12. Beat the Spread -10812
13. You can Say 11 - 10929
14. How old are you - 11219
15. Is this the easiest problem? - 11479
16. Cost cutting - 11727
17. A Benevolent Josephus - 1176
18. Egypt -11854
19. Brick game -11875
20. Multiple of 17 – 11879
21. A change in thermal unit 11984
22. Packing for holiday - 12372
23. Zapping – 12468
24. Hardest problem ever - 12478
25. Hazi-e-akbar12577
26. Zero or One - 12646
27. Little Master -12704
28. GCD the largest 12708
29. Ugly Number – 136
30. Counting weekend days -1727
31. Train Swapping – 299
32. The Collatz-Sequence – 694
33. Ecological bin packing - 102
34. **Problem name:** Hashmat the Brave Warrior -10055

**C Solution:**

#include<stdio.h>

int main() {

long long a, b;

while(scanf("%lld %lld", &a, &b) == 2) {

if(a > b) printf("%lld\n", a-b);

else printf("%lld\n", b-a);

}

return 0;

}

**Assembly Solution:**

;10055 - Hashmat the Brave Warrior

.MODEL SMALL

.STACK 100H

.CODE

MAIN PROC

START:

CALL SCAN

MOV BH, DL ;move first value in bh

CALL SCAN

MOV BL, DL ;move second value in bl

SUB BH, BL ;SUBTRACT THE INPUT VALUES

MOV AX, 0 ;CLEAR AX

MOV AL, BH ;MOVE SUBTRACTION TO AL FOR PRINT PROC

CMP AL, 0

JG P ;IF SUBTRACTION IS NOT GREATER THAN ZERO

NEG AL ;NEGATE SUBTRACTION

P:

CALL PRINT

MOV AH, 2

MOV DL, 10 ;PRINT NEW LINE

INT 21H

MOV DL, 13 ;PRINT CARRIGE RETURN

INT 21H

JMP START ;TAKE INPUT AGAIN

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

1. **Problem name:** Back to High School Physics -10071

**C Solution:**

#include<stdio.h>

int main() {

long long v, t;

while(scanf("%lld %lld", &v, &t) == 2)

printf("%lld\n", 2\*v\*t);

return 0;

}

**Assembly Solution:**

;10071 - Back to High School Physics

ORG 100h

MAIN PROC

START:

CALL SCAN ;TAKE V

MOV BH, DL

CALL SCAN ;TAKE T

MOV BL, DL

MOV AX, 0 ;CLEAR AX

MOV AL, BH

IMUL BL ;AX WILL STORE V\*T

MOV BL, 2

IMUL BL ;AX WILL STORE 2VT

CALL PRINT

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

1. **Problem name:** Pizza Cutting -10079

**C Solution:**

#include<stdio.h>

int main() {

long long int k;

while(scanf("%lld", &k) == 1 && k >= 0) {

printf("%lld\n", k\*(k+1)/2+1);

}

return 0;

}

**Assembly Solution:**

;10079 - Pizza Cutting

;UNLIKE THE UVA PROBLEM

;THIS PROGRAM IS TERMINATE ON INPUTTING ZERO

ORG 100h

MAIN PROC

START:

CALL SCAN ;INPUT NUMBER OF LINES

CMP DL, 0 ;IF INPUT IS ZERO

JE ENDMAIN ;TERMINATE

MOV CX, 0 ;CLEAR CX

MOV CL, DL

MOV AX, 1 ;MOVED 1 TO AX

SUM:

ADD AL, CL

LOOP SUM

CALL PRINT

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START ;TAKE INPUT AGAIN

ENDMAIN:

RET

ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

1. **Problem name:** The Land Of Justice - 10499

**C Solution:**

#include<stdio.h>

using namespace std;

int main(){

long int n,r;

while(scanf("%ld",&n)==1){

if(n<0) break;

if(n==1) r=0;

else r=n\*25;

printf("%ld%%\n",r);

}

return 0;

}

**Assembly Solution:**

;10499 - The Land of Justice

ORG 100H

.DATA

nl DB '%', 10, 13, '$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

MOV BX, 25

MOV CX, 4

TESTCASE:

CALL SCAN

CMP AX, 0

JL ENDCASE ;If negative, terminate

CMP AX, 1

JG CALC

MOV AX, 0

JMP PRINTRES

CALC:

IMUL BX

PRINTRES:

CALL PRINT

MOV AH, 9

LEA DX, nl

INT 21h

JMP TESTCASE

ENDCASE:

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

1. **Problem name:** Big Chocolate-10970

**C Solution:**

#include<iostream>

using namespace std;

int main(){

long m,n;

while(cin>>m>>n){

cout<<(m\*n)-1<<endl;

}

return 0;

}

**Assembly Solution:**

;10970 - Big Chocolate

ORG 100h

MAIN PROC

START:

CALL SCAN ;INPUT M

MOV BH, DL

CALL SCAN ;INPUT N

MOV AX, 0

MOV AL, DL

IMUL BH

DEC AL

CALL PRINT ;OUTPUT RESULT

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

JMP START ;TAKE INPUT AGAIN

ENDMAIN:

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

1. **Problem name:** Relational Operator -11172

**C Solution:**

#include<stdio.h>

int main() {

int T, x, y;

scanf("%d", &T);

while(T--) {

scanf("%d %d", &x, &y);

if(x < y) puts("<");

else if(x > y) puts(">");

else puts("=");

}

return 0;

}

**Assembly Solution:**

;11172 - Relational Operator

ORG 100h

MAIN PROC

CALL SCAN ;INPUT TESTCASE

MOV CX, 0 ;CLEAR CX

MOV CL, DL

MOV AH, 2

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

TESTCASE:

PUSH CX

CALL SCAN ;TAKE FIRST INPUT

MOV BH, DL

CALL SCAN ;TAKE SECOND INPUT

MOV AH, 2

MOV BL, DL

CMP BH, BL

JG GREATER

JL LESS

EQUAL: ;PRINT EQUAL

MOV DL, '='

INT 21H

JMP NEWLINE

GREATER:

MOV DL, '>'

INT 21H

JMP NEWLINE

LESS:

MOV DL, '<'

INT 21H

NEWLINE:

MOV DL, 10

INT 21H

MOV DL, 13

INT 21H

POP CX

LOOP TESTCASE

RET

MAIN ENDP

;a procedure that read an integer value

;of one or more digit

;input is terminated by both space and new line

;the inputed integer will be present in dl

SCAN PROC

MOV DX, 0

INPUT:

MOV AH, 1

INT 21h

CMP AL, ' '

JE END

CMP AL, 13

JE END

PUSH AX

MOV AL, 10

MUL DL

MOV DL, AL

POP AX

SUB AL, '0'

ADD DL, AL

JMP INPUT

END:

RET

SCAN ENDP

;a procedure that prints an integer

;consisting of one or more digits

;the integer must be present in ax

PRINT PROC

XOR CX, CX

LOOP1:

CWD

MOV BX, 10

IDIV BX

PUSH DX

INC CX

CMP AX, 0

JG LOOP1

LOOP2:

POP DX

ADD DX, '0'

MOV AH, 2

INT 21h

LOOP LOOP2

RET

PRINT ENDP

1. **Problem name:** Horror Dash-11799

**C Solution:**

#include <stdio.h>

int main() {

int t, n, Case = 0;

scanf("%d", &t);

while(t--) {

scanf("%d", &n);

int max = 0, c;

while(n--) {

scanf("%d", &c);

if(max < c)

max = c;

}

printf("Case %d: %d\n", ++Case, max);

}

return 0;

}

**Assembly Solution:**

;11799 - Horror Dash

ORG 100H

.DATA

n DW ?

max DW ?

case DW ?

c DW ?

msg DB 'CASE $'

.CODE

MAIN PROC

START:

MOV AX,@DATA

MOV DS,AX

CALL SCAN

XOR CX,CX

MOV CX,AX

MOV CASE,0

CASE1:

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

PUSH CX

MOV MAX,0

CALL SCAN

MOV CX,AX

CASE2:

CALL SCAN

MOV c,AX

CMP AX,MAX

JG EQUAL

LOOP CASE2

JMP NEXT

EQUAL:

MOV AX,c

MOV MAX,AX

LOOP CASE2

NEXT:

POP CX

INC CASE

LEA DX,MSG

MOV AH,9

INT 21h

MOV AX,CASE

CALL PRINT

MOV AH,2

MOV DX,':'

INT 21h

MOV DX,' '

INT 21h

MOV AX,MAX

CALL PRINT

LOOP CASE1

MOV AH,2

MOV DX,10

INT 21h

MOV DX,13

INT 21h

JMP START

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUT

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUT:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUT

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

1. **Problem name: Bafana Bafana -11805**

**C Solution:**

#include <stdio.h>

int main() {

int t, test = 0, N, K, P;

scanf("%d", &t);

while(t--) {

scanf("%d %d %d", &N, &K, &P);

printf("Case %d: %d\n", ++test, (K+P-1)%N+1);

}

return 0;

}

**Assembly Solution:**

;11805 - Bafana Bafana

ORG 100H

.DATA

case1 DB 'Case $'

case2 DB ': $'

nl DB 10, 13, '$'

.CODE

MAIN PROC

MOV AX, @DATA

MOV DS, AX

CALL SCAN

MOV CX, AX

MOV BX, 1

TESTCASE:

PUSH CX

MOV AH, 9

LEA DX, case1

INT 21h

MOV AX, BX ;Case no

CALL print

PUSH BX

MOV AH, 9

LEA DX, case2

INT 21h

CALL scan

MOV BX, AX ;n

CALL scan

MOV CX, AX ;k

CALL scan ;p

ADD AX, CX ;k+p

CWD

IDIV BX

MOV AX, DX ;(k+p)%n

OR AX, AX

JNZ NOTZERO

ADD AX, BX

NOTZERO:

CALL PRINT

MOV AH, 9

LEA DX, nl

INT 21h

POP BX

INC BX

POP CX

LOOP TESTCASE

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDINPUT

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

1. **Problem name: Peter’s Smoke - 10346**

**C Solution:**

#include <stdio.h>

int main() {

int n, k;

while(scanf("%d %d", &n, &k) == 2) {

int sum = n, used = n;

while(used >= k) {

n = used/k;

used = used%k + n;

sum += n;

}

printf("%d\n", sum);

}

return 0;

}

**Assembly Solution:**

;10346 - Peter's Smoke

ORG 100H

MAIN PROC

TESTCASE:

CALL SCAN

CMP AX, -1

JE ENDCASE:

MOV BX, AX

CALL SCAN

XCHG BX, AX ;AX = n, BX = k

MOV CX, AX ;CX = total cigarette

CALCULATE:

CWD

IDIV BX

CMP AX, 0

JZ ENDCULATE

ADD CX, AX

ADD AX, DX

JMP CALCULATE

ENDCULATE:

MOV AX, CX

CALL PRINT

MOV AH, 2

MOV DL, 10

INT 21h

MOV DL, 13

INT 21h

JMP TESTCASE

ENDCASE:

RET

ENDP

;A procedure that reads a 16 bit signed input

;and store that in AX

SCAN PROC

;Backup register values in stack

PUSH BX

PUSH CX

PUSH DX

;Clear register values

XOR BX, BX

XOR CX, CX

;Read first character

MOV AH, 1

INT 21H

;Check if it is a sign or digit

CMP AL, '-'

JE NEGATIVE

CMP AL, '+'

JE POSITIVE

JMP INPUTSCAN

NEGATIVE:

;Store that it is negative number in CX

MOV CX, 1

POSITIVE:

;Take a digit input if first input is sign

INT 21H

INPUTSCAN:

;Convert the digit ASCII to number

AND AX, 000FH

;As multiplication erases value in AX

;backup the digit to stack

PUSH AX

;Multiply previous value by 10 and add new value

MOV AX, 10

MUL BX

;Pop new digit from stack

POP BX

ADD BX, AX

;Read digit repeatedly until space or carriage return read

MOV AH, 1

INT 21H

CMP AL, ' '

JE ENDINPUT

CMP AL, -1

JE ENDFILE

CMP AL, 13

JE CARRIAGERETURN

JMP INPUTSCAN

CARRIAGERETURN:

;If last input is carriage return, print a new line

MOV AH, 2

MOV DL, 10

INT 21H

;Store the positive input to AX

ENDINPUT:

MOV AX, BX

;Check if the value is negative

CMP CX, 0

JE ENDSCAN

NEG AX

ENDFILE:

MOV AX, -1

ENDSCAN:

;Restore register values from stack

POP DX

POP CX

POP BX

RET

ENDP

PRINT PROC

;Backup register values in stack

PUSH AX

PUSH BX

PUSH CX

PUSH DX

;Check if Ax is positive or negative

CMP AX, 0

JGE INIT

PUSH AX

MOV DL, '-'

MOV AH, 2

INT 21H

POP AX

NEG AX

INIT:

XOR CX, CX ;Clear CX. Holds number of digits

MOV BX, 10 ;Holds divisor

DIGITIFY:

CWD ;Clear DX

DIV BX

PUSH DX ;Push last digit to stack

INC CX

;Check if the quotient is zero

CMP AX, 0

JNZ DIGITIFY

;Pop and print

MOV AH, 2

PRINTLOOP:

POP DX

OR DL, 30H ;Convert to ASCII

INT 21H

LOOP PRINTLOOP

;Restore register values from stack

POP DX

POP CX

POP BX

POP AX

RET

ENDP

1. **Problem name: F91 -10696**

**C Solution:**

#include<stdio.h>

int main() {

int n;

while(scanf("%d", &n) == 1 && n) {

printf("f91(%d) = %d\n", n, n > 100 ? n-10 : 91);

}

return 0;

}

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name:** Odd Sum -10783

**C Solution:**

#include<stdio.h>

int main() {

int T, Case = 0;

scanf("%d", &T);

while(T--) {

int i, sum = 0, a, b;

scanf("%d %d", &a, &b);

if((a&1) == 0) a++;

for(i = a; i <= b; i += 2)

sum += i;

printf("Case %d: %d\n", ++Case, sum);

}

return 0;

}

1. **Problem name:** Beat the Spread -10812

**C Solution:**

#include <stdio.h>

int main() {

int n, s, d, a, b;

scanf("%d", &n);

while(n--) {

scanf("%d %d", &s, &d);

a = (s+d);

b = (s-d);

if(a < 0 || b < 0 || a%2 == 1 || b%2 == 1)

puts("impossible");

else

printf("%d %d\n", a/2, b/2);

}

return 0;

}

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: You can Say 11-10929**

**C Solution:**

#include <stdio.h>

#include <string.h>

int main() {

char s[1001];

while(scanf("%s", s) == 1) {

if(strcmp(s, "0") == 0)

break;

int i, l = strlen(s), sum = 0;

for(i = l-1; i >= 0; i -= 2)

sum += s[i] - '0';

for(i = l-2; i >= 0; i -= 2)

sum -= s[i] - '0';

printf("%s ", s);

if(sum%11 == 0)

puts("is a multiple of 11.");

else

puts("is not a multiple of 11.");

}

return 0;

}

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |

1. **Problem name: 11805**

**C Solution:**

**Assembly Solution:**

|  |  |
| --- | --- |
|  |  |