**Lab 3a.s**

/\* DO NOT MODIFY THIS --------------------------------------------\*/

.text

.global WelcomePrompt

.extern iprintf

.extern cr

.extern value

.extern getstring

/\*----------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* General Information \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* File Name: Lab3a.s \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Names of Students: Darius Fang and Marlene Gong \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* IDs: 1570975 and 1572719 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Date: March 11, 2020 \*\*/

/\* General Description: Subroutine will prompt user to enter numbers \*\*/

/\* from the keyboard \*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

WelcomePrompt:

/\*Write your program here\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sub.l #40, %a7 /\* preserve registers ... \*/

movem.l %a2-%a5/%d2-%d7, (%a7) /\* ...except for stack pointer \*/

move.l #0x43000000, %a2 /\* set a2 as memory location of values to be analyzed \*/

clr.l %d5 /\* clear d5 \*/

pea Greeting /\* push welcome string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

addq.l #4, %sp /\* clean up stack \*/

move.l #0, %d3 /\* clear d3 \*/

get\_entry:

move.l #0, %d5 /\* move 0 to d5 \*/

pea Entries /\* push number of entries string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

addq.l #4, %sp /\* clean up stack \*/

jsr getstring /\* jump to getstring subroutine, store user input in d0 \*/

move.l %d0, %d2 /\* copy user input into d2 \*/

move.l %d2, -(%sp) /\* push user input onto stack \*/

jsr value /\* jump to value subroutine, display user input \*/

add.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

move.l %d2, 48(%sp) /\* replace 0xEEEEEEEE (from main program) with number of entries \*/

cmp.l #3, %d2 /\* compare number of entries to 3 \*/

blt error\_input /\* if less than 3, branch to error\_input \*/

cmp.l #15, %d2 /\* otherwise, compare number of entries to 15 \*/

ble get\_div /\* if less than or equal to 15, branch to get\_div \*/

bra error\_input /\* otherwise, branch to error\_input \*/

get\_div:

move.l #1, %d5 /\* move 1 to d5 \*/

move.l %d2, %d4 /\* copy number of entries to d4 \*/

sub.l #1, %d4 /\* subtract number of entries by 1 \*/

pea Divisor /\* push divisor string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

addq.l #4, %sp /\* clean up stack \*/

jsr getstring /\* jump to getstring subroutine, store user input in d0 \*/

move.l %d0, %d2 /\* copy user input into d2 \*/

move.l %d2, -(%sp) /\* push user input onto stack \*/

jsr value /\* jump to value subroutine, display user input \*/

add.l #4, %sp /\* clean up stack \*/

move.l %d2, 44(%sp) /\* replace 0xDDDDDDDD (from main program) with divisor \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

cmp.l #2, %d2 /\* compare divisor to 2 \*/

blt error\_input /\* if less than 2, branch to error\_input \*/

cmp.l #5, %d2 /\* otherwise, compare divisor to 5 \*/

ble get\_posNum /\* if less than or equal to, branch to get\_posNum \*/

bra error\_input /\* otherwise, branch to error\_input \*/

get\_posNum:

move.l #2, %d5 /\* move 2 to d5 \*/

pea posNumber /\* push positive number string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

addq.l #4, %sp /\* clean up stack \*/

jsr getstring /\* jump to getstring subroutine, store user input in d0 \*/

move.l %d0, %d2 /\* copy user input into d2 \*/

move.l %d2, -(%sp) /\* push user input onto stack \*/

jsr value /\* jump to value subroutine, display user input \*/

addq.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

cmp.l #0, %d2 /\* compare number to 0 \*/

ble error\_input /\* if less than or equal to 0, branch to error\_input \*/

move.l %d2, (%a2)+ /\* otherwise, copy number to memory location of values to be analyzed, increment a2 \*/

sub.l #1, %d4 /\* subtract number of entries by 1 \*/

bne get\_posNum /\* if number of entries doesn't equal zero, branch to get\_posNum \*/

bra get\_lastNum /\* otherwise, branch to get\_lastNum \*/

get\_lastNum:

move.l #3, %d5 /\* move 3 to d5 \*/

pea lastNumber /\* push last number string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

addq.l #4, %sp /\* clean up stack \*/

jsr getstring /\* jump to getstring subroutine, store user input in d0 \*/

move.l %d0, %d2 /\* copy user input into d2 \*/

move.l %d2, -(%sp) /\* push user input onto stack \*/

jsr value /\* jump to value subroutine, display user input \*/

addq.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

cmp.l #0, %d2 /\* compare last number to 0 \*/

ble error\_input /\* if less than or equal to 0, branch to error\_input \*/

move.l %d2, (%a2)+ /\* otherwise, copy number to memory location of values to be analyzed, increment a2 \*/

bra end /\* branch to end \*/

error\_input:

pea Invalid /\* push invalid entry string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

addq.l #4, %sp /\* clean up stack \*/

cmp.l #0, %d5 /\* compare d5 to 0 \*/

beq get\_entry /\* if equal, branch to get\_entry \*/

cmp.l #1, %d5 /\* otherwise, compare d5 to 1 \*/

beq get\_div /\* if equal, branch to get\_div \*/

cmp.l #2, %d5 /\* otherwise, compare d5 to 2 \*/

beq get\_posNum /\* if equal, branch to get\_posNum \*/

cmp.l #3, %d5 /\* otherwise, compare d5 to 3 \*/

beq get\_lastNum /\* if equal, branch to get\_lastNum \*/

end:

movem.l (%a7), %a2-%a5/%d2-%d7 /\* restore registers \*/

add.l #40, %a7 /\* stack pointer points at return address \*/

rts /\* return to main program \*/

/\*End of Subroutine \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

.data

/\*All Strings placed here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Greeting:

.string "Welcome to Wing's Stats Program"

Entries:

.string "Please enter the number(3min-15max) of entries followed by 'enter'"

Divisor:

.string "Please enter the divisor(2min-5max) followed by 'enter'"

posNumber:

.string "Please enter a number(positive only)"

lastNumber:

.string "Please enter the last number(positive only)"

Invalid:

.string "Invalid entry, please enter proper value."

/\*End of Strings \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Lab3b.s**

/\* DO NOT MODIFY THIS --------------------------------------------\*/

.text

.global Stats

.extern iprintf

.extern cr

.extern value

.extern getstring

/\*----------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* General Information \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* File Name: Lab3b.s \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Names of Students: Darius Fang and Marlene Gong \*\*\*\*\*\*\*/

/\* IDs: 1570975 and 1572719 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Date: March 11, 2020 \*\*/

/\* General Description: Subroutine will find the min, max, \*\*/

/\* mean, how many numbers were divisible by the divisor \*\*/

/\* and what are they from the numbers entered \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Stats:

/\*Write your program here\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sub.l #40, %a7 /\* preserve registers ... \*/

movem.l %a2-%a5/%d2-%d7, (%a7) /\* ...except for stack pointer \*/

/\* substitute values for labels \*/

.equ Min, 0

.equ Max, 4

.equ Mean, 8

.equ Divisible, 12

/\*

a2: array index (values to be analyzed), a3: results

d2: divisable

d3: entries

d4: dummy variable

d5: number of div

d6: counter

\*/

move.l %d3, %d6 /\* copy the number of entries to a counter \*/

clr.l %d5 /\* clear d5 \*/

clr.l Mean(%a3) /\* clear contents of 0x43100008 \*/

move.l (%a2), %d4 /\* copy first entry to d4 \*/

move.l %d4, Min(%a3) /\* copy entry into 0x43100000 \*/

move.l %d4, Max(%a3) /\* copy entry into 0x43100004

Loop:

move.l (%a2)+, %d4 /\* copy entry to d4, increment pointer to next entry \*/

cmp.l Min(%a3), %d4 /\* compare entry to contents of 0x43100000 \*/

blt min /\* if less than, branch to min \*/

Return1:

cmp.l Max(%a3), %d4 /\* otherwise, compare entry to contents of 0x43100004 \*/

bgt max /\* if greater than, branch to max \*/

Return2:

add.l %d4, Mean(%a3) /\* otherwise, add entry to contents of 0x43100008 \*/

divu.w %d2, %d4 /\* divide entry by divisor \*/

lsr.l #8, %d4 /\* shift result by 8 bits \*/

lsr.l #8, %d4 /\* shift result by 8 bits \*/

beq div /\* if remainder is equal to 0, branch to div \*/

Return3:

sub.l #1, %d6 /\* decrement counter \*/

bne Loop /\* if counter isn’t 0, branch to Loop \*/

bra end /\* otherwise, branch to end \*/

min:

move.l %d4, Min(%a3) /\* copy d4 into 0x43100000 \*/

bra Return1 /\* branch to Return 1 \*/

max:

move.l %d4, Max(%a3) /\* copy d4 into 0x43100004 \*/

bra Return2

div:

move.l %d5, %d4 /\* copy number of divisible entries to d4 \*/

muls.w #4, %d4 /\* multiply d4 by 4 \*/

move.l -(%a2), Divisible(%a3, %d4) /\* point at previous entry and copy to index in 0x4310000C \*/

add.l #4, %a2 /\* point at next entry \*/

add.l #1, %d5 /\* add 1 to number of divisible entries \*/

bra Return3

end:

move.l Mean(%a3), %d4 /\* copy contents of 0x43100008 to d4 \*/

divu.l %d3, %d4 /\* divide d4 by the number of entries \*/

move.l %d4, Mean(%a3) /\* copy results into 0x43100008 \*/

move.l %d5, 52(%sp) /\* replace 0xFFFFFFFF (from main program) with number of divisible entries\*/

movem.l (%a7), %a2-%a5/%d2-%d7 /\* restore registers \*/

add.l #40, %a7 /\* stack pointer points at return address \*/

rts /\* return to main program \*/

/\*End of Subroutine \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

.data

/\*All Strings placed here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*End of Strings \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Lab3c.s**

/\* DO NOT MODIFY THIS --------------------------------------------\*/

.text

.global Display

.extern iprintf

.extern cr

.extern value

.extern getstring

/\*----------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* General Information \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* File Name: Lab3c.s \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Names of Students: Darius Fang and Marlene Gong \*\*\*\*\*\*\*\*\*/

/\* IDs: 1570975 and 1572719 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Date: March 11, 2020 \*\*/

/\* General Description: Subroutine will display the min, max, \*\*/

/\* mean, how many numbers were divisible by the divisor \*\*/

/\* and what are they on MTTTY \*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Display:

/\*Write your program here\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

sub.l #40, %a7 /\* preserve registers ... \*/

movem.l %a2-%a5/%d2-%d7, (%a7) /\* ...except for stack pointer \*/

pea numEntries /\* push number of entries string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

addq.l #4, %sp /\* clean up stack \*/

move.l %d3, -(%sp) /\* push number of entries onto stack \*/

jsr value /\* jump to value subroutine, display number of entries \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

add.l #4, %sp /\* clean up stack \*/

loop:

move.l (%a2)+, %d5 /\* copy entry into d5, increment to next entry \*/

move.l %d5, -(%sp) /\* push entry onto stack \*/

jsr value /\* jump to value subroutine, display entry \*/

add.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

sub.l #1, %d3 /\* subtract 1 from the number of entries \*/

bne loop /\* if number of entries isn’t 0, branch to loop \*/

movea.l #0x43100000, %a2 /\* copy 0x43100000 to a2 \*/

pea minNum /\* push min number string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

move.l (%a2)+, %d3 /\* copy min number to d3, increment to max number \*/

move.l %d3, -(%sp) /\* push min number onto stack \*/

jsr value /\* jump to value subroutine, display min number \*/

add.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

pea maxNum /\* push max number string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

move.l (%a2)+, %d3 /\* copy max number to d3, increment to mean number \*/

move.l %d3, -(%sp) /\* push max number onto stack \*/

jsr value /\*jump to value subroutine, display max number \*/

add.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

pea meanNum /\* push mean number string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

move.l (%a2)+, %d3 /\* copy mean number to d3, increment a2 \*/

move.l %d3, -(%sp) /\* push mean number onto stack \*/

jsr value /\* jump to value subroutine, display mean number \*/

add.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

pea ThereAre /\* push “There are” string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

move.l %d4, -(%sp) /\* push number of divisible entries onto stack \*/

jsr value /\* jump to value subroutine, display number of divisible entries \*/

add.l #4, %sp /\* clean up stack \*/

pea divisible /\* push “ number(s) divisible by “ string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

move.l %d2, -(%sp) /\* push divisor onto stack \*/

jsr value /\* jump to value subroutine, display divisor \*/

add.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

pea TheyAre /\* push “They are” string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

divLoop:

pea space /\* push “ “ string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

move.l (%a2)+, %d3 /\* copy divisible number to d3, increment to next divisible number \*/

move.l %d3, -(%sp) /\* push divisible number onto stack \*/

jsr value /\* jump to value subroutine, display divisible number \*/

add.l #4, %sp /\* clean up stack \*/

sub.l #1, %d4 /\* subtract 1 from number of divisible entries \*/

bne divLoop /\* if number of divisible entries isn’t 0, branch to divLoop \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

pea end /\* push program end string onto stack \*/

jsr iprintf /\* jump to iprintf subroutine, print string \*/

add.l #4, %sp /\* clean up stack \*/

jsr cr /\* jump to cr subroutine, generate carriage return and linefeed \*/

movem.l (%a7), %a2-%a5/%d2-%d7 /\* restore registers \*/

add.l #40, %a7 /\* stack pointer points at return address \*/

rts /\* return to main program \*/

/\*End of Subroutine \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

.data

/\*All Strings placed here \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

numEntries:

.string "The number of entries was "

minNum:

.string "Min number = "

maxNum:

.string "Max number = "

meanNum:

.string "Mean number = "

ThereAre:

.string "There are "

divisible:

.string " number(s) divisible by "

TheyAre:

.string “They are”

space:

.string “ “

end:

.string "Program ended"

/\*End of Strings \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/