Assignment 2

CSCI131

Name: James Marino

Username: jm617

Student Number: 4720994

Table Of Contents

1. Executive Summary

This report aims to show calculations with a large set of data finding the mean, mode, minimum and maximum values. The programs to calculate this data were written in the C programming language and the PDP-11 Assembly Language.

2. Introduction

Both sets of code are commented throughout, heavily focusing on the PDP-11 Assembly providing details and a breakdown of steps of the code and what each one does. An outline of each function in Assembly, Mean, Mode and MaxMin are provided.

3. Body

3.1 C Program

Code for finding Mean, Mode and MinMax Values is below

```c
#include <stdio.h>
#include <limits.h>

const int DATA_SIZE = 250;

const int DATA[] = {
	1, 4, 8, 20, 4, 39, 21, 23, 16, 5,
	3, 7, 63, 1, 42, 4, 35, 3, 29, 17,
	8, 24, 54, 94, 28, 52, 63, 28, 59, 47,
	43, 78, 34, 35, 44, 39, 32, 63, 67, 63,
	59, 45, 83, 68, 39, 68, 27, 60, 59, 49,
	73, 61, 54, 52, 50, 74, 58, 43, 64, 64,
	69, 54, 42, 52, 50, 50, 61, 81, 76, 41,
	30, 53, 68, 66, 57, 59, 48, 71, 33, 75,
	35, 53, 48, 60, 65, 52, 60, 37, 44, 48,
	54, 76, 71, 27, 50, 64, 98, 65, 55, 60,
	75, 60, 55, 49, 53, 52, 52, 35, 53, 24,
	27, 79, 69, 44, 42, 51, 46, 66, 50, 52,
	71, 59, 57, 49, 59, 56, 51, 71, 84, 43,
	72, 46, 76, 67, 73, 52, 65, 59, 27, 58,
	37, 47, 65, 71, 96, 45, 53, 55, 59, 74,
	34, 72, 46, 41, 59, 80, 36, 61, 44, 42,
	53, 40, 67, 48, 54, 46, 71, 75, 46, 60,
	43, 58, 63, 54, 62, 89, 54, 75, 66, 72,
	66, 66, 74, 34, 50, 61, 60, 56, 75, 88,
	61, 96, 44, 71, 62, 79, 60, 65, 57, 60,
	54, 56, 65, 52, 53, 81, 52, 59, 67, 39,
	53, 70, 67, 30, 81, 85, 66, 41, 56, 68,
	90, 56, 72, 63, 58, 69, 63, 54, 60, 66,
	69, 50, 35, 63, 38, 28, 74, 87, 66, 73,
	51, 62, 51, 64, 61, 63, 86, 62, 71, 77,
};

void mean()
{
	int total = 0, i = 0, mean = 0;

	for (i = 0; i < DATA_SIZE; i++) {
		total += DATA[i];
	}

	mean = total / DATA_SIZE;

	printf("Mean : %d\n", mean);
}
```

```
void mode()
{
	int i;
	int dmode[101];

	for (i = 0; i <= 100; i++) {
		dmode[i] = 0;
	}

	for (i = 0; i < DATA_SIZE; i++) {
		int value = DATA[i];
		dmode[value]++;
	}

	int maxndx = 0, maxval= 0;

	for (i = 0; i <= 100; i++) {
		if (dmode[i] > maxval) {
			maxval = dmode[i];
			maxndx = i;
		}
	}

	printf("Mode : %d\n", maxndx);
}

void maxAndMin()
{
	int max = INT_MIN;
	int min = INT_MAX;
	int i = 0, j = 0, current = 0;

	for (i = 0; i < DATA_SIZE; i++) {
		current = DATA[i];

		for (j = 0; j < DATA_SIZE; j++) {

			// Test Max
			if (current > max) {
				max = current;
			}

			// Test Min
			if (current < min) {
				min = current;
			}

		}
	}

	printf("Max : %d\n", max);
```

```
    printf("Min : %d\n", min);
}

int main()
{
    mean();
    mode();
    maxAndMin();

   return 0;
}
```

Straight forward C Code, follows similar logical concepts as assembly in PDP-11.

3.2 PDP-11 Assembly Program

Questions:

For each function, explain how you have encoded the loop constructs in assembly language and how you are accessing the array elements.

- Loops where done using the “subtract one and branch if counter is 0” OP code known as *sob*.
  - The array length count was used as the counter for the loop and as a result, the counter was used in the parameters of the sob operation
  - OP Code was used as followed: sob *counterRegister, labelOfStartOfLoop*

- Arrays were traversed by using the address of the array in memory and the fact that a “word”, each element was 2 bytes long.
- Using this, we can add an octal “2” value to the address of the pointer thus accessing the next element in the array, or the next section in the PDP-11’s memory.

Mean Function Explanation:
The mean function loops through the size of the array, each iteration adding (add) the current value of the array to a global counter in a register for all elements in the array.

There is then a division between the amount of elements and the overall count. this is done using the (div) OP Code

Mode Function Explanation:
The mode function took a value of array *index i* in an outer loop and stored this value in memory. An inner loop to this loop traversed through the same array and counted how many times matches occurred (beq).

Each iterations matches were stored only if the existing count of matches was greater than the current count of matches (bgt).

Max Min Function Explanation:
The minmax function set initial starting values having the max value set to smallest octal number and min number to largest octal number that the PDP-11 could process.

The array was then traversed and current min and max values where checked with the existing “highest so far” values.

Code for PDP-11 Assembly is below. Note I/O Libraries have **not** been included.

```
.origin 2000
; ------------------ Start Application ------------------
;
; Function: Calculate the mean
meanc:
      ;
      ; Clear registers before use
      clr r0
      clr r1
      clr r2
      clr r3
      clr r4
      clr r5
      ; Load size of array in r3
      mov #datasize, r4
      ; Load the address of the array into r0
      mov #dataset, r0
      ;
      ; Start Loop
      loop:
            ; Move the current value of r0 into r2
            mov @r0, r2
            ;
            ; Work with the current value
            ; Self add to r3
            add r2, r3
            ;
            ; Add 2 bytes to the array pointer
            add #2, r0
            ; subtract and branch if counter is equal to 0
            ; i.e. go back to the loop statement if not done
            sob r4, loop
      ;
      ; Divide by total addition of numbers
      clr r0
      ; arg1: the value to be divided
      ; arg2: where the remainder will end up
      mov r3, r1
      ; arg1: Divide what you are going to divide by
      ; arg2: where the quotiant will be stored
      div #datasize, r0
      ;
      ; Move the register holding final mean value
      ; Into mean variable
      mov r0, meanv
      return
;
; Function: Print the mean
mean:
      ; Process data
      call meanc
      ; Print out values
      writeline
      msgmean
      itoa
      meanv
      numbuf
      writeline
      numbuf
      return
;
; Function: Calculate the mode
modec:
      ; Set highest value in memory
      highestvalue=0
      ;
      ; Clear registers before use
      clr r0
      clr r1
      clr r2
      clr r3
      clr r4
      clr r5
      ;
      ; ----- Register Layout -----
      ; r0 and r1: Load the address of the array
      mov #dataset, r0
      mov #dataset, r1
      ;
      ; r2 and r3: Load size of array (as counters)
      mov #datasize, r2
      mov #datasize, r3
      ;
      ; r4: Current count of equal values in array
      ; r5: Highest count of values
      ; highestvalue: Highest Value so far --> stored in ram
      ;
      ; Loop 1 (All numbers in array)
      loop1:
            ;
            ; Clear the previous loops equal counter r4
            clr r4
            ;
            ; Loop 2: Again (All numbers in array)
            loop2:
                  ;
                  ; ---------- Loop 2 Contents ------------------
                  ;
                  ; Branch if current loop 1 array value is equal
                  ; to current list in loop 2 is traversing
                  cmp @r0, @r1
                        ; Branch of equal
```

```
                    beq equalinc
                    bne continueequalinc
               ; Function if equal
               equalinc:
                    ; Add to the current count
                    inc r4
               ; Not Equal
               continueequalinc:
               ;
               ; ---------- Loop 2 Contents ------------------
               ;
               ; Add 2 bytes to the array pointer
               add #2, r1
               ; subtract and branch if counter is equal to 0
               ; i.e. go back to the loop statement if not done
               sob r3, loop2
          ;
          ; ---------- Loop 1 Contents ------------------
          ;
          ; Compare r4 (current count) with highest value
          cmp r4, r5
               bgt storehigh
               blt continuestorehigh
          ;
          ; Function if greater than
          storehigh:
               ; Move value of r4 into r5
               mov r4, r5
               ; Update actual value
               mov @r0, highestvalue
          ; Continue function
          continuestorehigh:
          ;
          ;
          ; ---------- Loop 1 Contents ------------------
          ;
          ;
          ; Reset r3
          mov #datasize, r3
          ; Reset r1 array to begining
          mov #dataset, r1
          ;
          ; Add 2 bytes to the array pointer
          add #2, r0
          ; subtract and branch if counter is equal to 0
          ; i.e. go back to the loop statement if not done
          sob r2, loop1
     ;
     ; Move highestvalue into buffer
     mov highestvalue, modev
     ;
     return
```

```
;
; Function: Print the mode
mode:
      ; Process data
      call modec
      ; Print out values
      writeline
      msgmode
      itoa
      modev
      numbuf
      writeline
      numbuf
      return
;
; Function: Calculate the max and min
maxandminc:
      ;
      ; Clear registers before use
      clr r0
      clr r1
      clr r2
      clr r3
      clr r4
      clr r5
      ;
      ; ----- Register Layout -----
      ; r0: Load the address of the array
      mov #dataset, r0
      ;
      ; r1: Load size of array (as counters)
      mov #datasize, r1
      ;
      ; r2: Current array value
      ; r3: Current smallest value
      ; r4: Current largest value
      ;
      ; Store pre loop max and min values (opposites)
      ; for initial compare
      mov #77777, r3
      mov #00000, r4
      ;
      ; Loop 1 (All numbers in array)
      loopmaxmin:
            ;
            ; ---------- Loop Contents -----------------
            ;
            ; Get current array value
            mov @r0, r2
            ;
            ; Compare the min value
            cmp r2, r3
```

```
                        ; Branch if less than
                        blt minstore
                        ;
                ; Compare max value
                cmp r2, r4
                        ; Branch greater than
                        bgt maxstore
                        ;
                ; Compare equal to
                cmp r2, r3
                        beq continue
                cmp r2, r4
                        beq continue
                        ;
                ; Compare in between
                cmp r2, r3
                        bgt continue
                cmp r2, r4
                        blt continue
                        ;
                ; Store min value
                minstore:
                        mov r2, r3
                        jmp continue
                ;
                ; Store max value
                maxstore:
                        mov r2, r4
                ;
                ; Continue
                continue:
                ;
                ; ---------- Loop Contents ------------------
                ;
                ; Add 2 bytes to the array pointer
                add #2, r0
                ; subtract and branch if counter is equal to 0
                ; i.e. go back to the loop statement if not done
                sob r1, loopmaxmin
        ;
        ; Store values into buffer
        mov r3, minv
        mov r4, maxv
        ;
        return
;
; Function: Print the max and min
maxandmin:
        ; Process data
        call maxandminc
        ; Print out values
        ; Max Values
```

```
      writeline
      msgmax
      itoa
      maxv
      numbuf
      writeline
      numbuf
      ;
      ; Newline
      writeline
      newline
      ;
      ; Min Values
      writeline
      msgmin
      itoa
      minv
      numbuf
      writeline
      numbuf
      ; Return
      return
;
; Function: int main()
application:
      ; Main Code
      call mean
      ;
      writeline
      newline
      ;
      call mode
      ;
      writeline
      newline
      ;
      call maxandmin
      ;
      exit
.origin 3000
;
; Strings
msgmean: .string "Mean  : "
msgmode: .string "Mode  : "
msgmax: .string "Max  : "
msgmin: .string "Min  : "
;
; Variables
meanv: .blkw 1
modev: .blkw 1
maxv: .blkw 1
minv: .blkw 1
```

```
;
; Data
dataset:
.word 1
.word 4
.word 10
.word 24
.word 4
.word 47
.word 25
.word 27
.word 20
.word 5
.word 3
.word 7
.word 77
.word 1
.word 52
.word 4
.word 43
.word 3
.word 35
.word 21
.word 10
.word 30
.word 66
.word 136
.word 34
.word 64
.word 77
.word 34
.word 73
.word 57
.word 53
.word 116
.word 42
.word 43
.word 54
.word 47
.word 40
.word 77
.word 103
.word 77
.word 73
.word 55
.word 123
.word 104
.word 47
.word 104
.word 33
.word 74
.word 73
```

```
.word 61
.word 111
.word 75
.word 66
.word 64
.word 62
.word 112
.word 72
.word 53
.word 100
.word 100
.word 105
.word 66
.word 52
.word 64
.word 62
.word 62
.word 75
.word 121
.word 114
.word 51
.word 36
.word 65
.word 104
.word 102
.word 71
.word 73
.word 60
.word 107
.word 41
.word 113
.word 43
.word 65
.word 60
.word 74
.word 101
.word 64
.word 74
.word 45
.word 54
.word 60
.word 66
.word 114
.word 107
.word 33
.word 62
.word 100
.word 142
.word 101
.word 67
.word 74
.word 113
```

```
.word 74
.word 67
.word 61
.word 65
.word 64
.word 64
.word 43
.word 65
.word 30
.word 33
.word 117
.word 105
.word 54
.word 52
.word 63
.word 56
.word 102
.word 62
.word 64
.word 107
.word 73
.word 71
.word 61
.word 73
.word 70
.word 63
.word 107
.word 124
.word 53
.word 110
.word 56
.word 114
.word 103
.word 111
.word 64
.word 101
.word 73
.word 33
.word 72
.word 45
.word 57
.word 101
.word 107
.word 140
.word 55
.word 65
.word 67
.word 73
.word 112
.word 42
.word 110
.word 56
```

```
.word 51
.word 73
.word 120
.word 44
.word 75
.word 54
.word 52
.word 65
.word 50
.word 103
.word 60
.word 66
.word 56
.word 107
.word 113
.word 56
.word 74
.word 53
.word 72
.word 77
.word 66
.word 76
.word 131
.word 66
.word 113
.word 102
.word 110
.word 102
.word 102
.word 112
.word 42
.word 62
.word 75
.word 74
.word 70
.word 113
.word 130
.word 75
.word 140
.word 54
.word 107
.word 76
.word 117
.word 74
.word 101
.word 71
.word 74
.word 66
.word 70
.word 101
.word 64
.word 65
```

```
.word 121
.word 64
.word 73
.word 103
.word 47
.word 65
.word 106
.word 103
.word 36
.word 121
.word 125
.word 102
.word 51
.word 70
.word 104
.word 132
.word 70
.word 110
.word 77
.word 72
.word 105
.word 77
.word 66
.word 74
.word 102
.word 105
.word 62
.word 43
.word 77
.word 46
.word 34
.word 112
.word 127
.word 102
.word 111
.word 63
.word 76
.word 63
.word 100
.word 75
.word 77
.word 126
.word 76
.word 107
.word 115
datasize=372
;
; IO Data
newline: .word 15
input: .blkw 100
output: .blkw 100
numbuf: .blkw 5
```

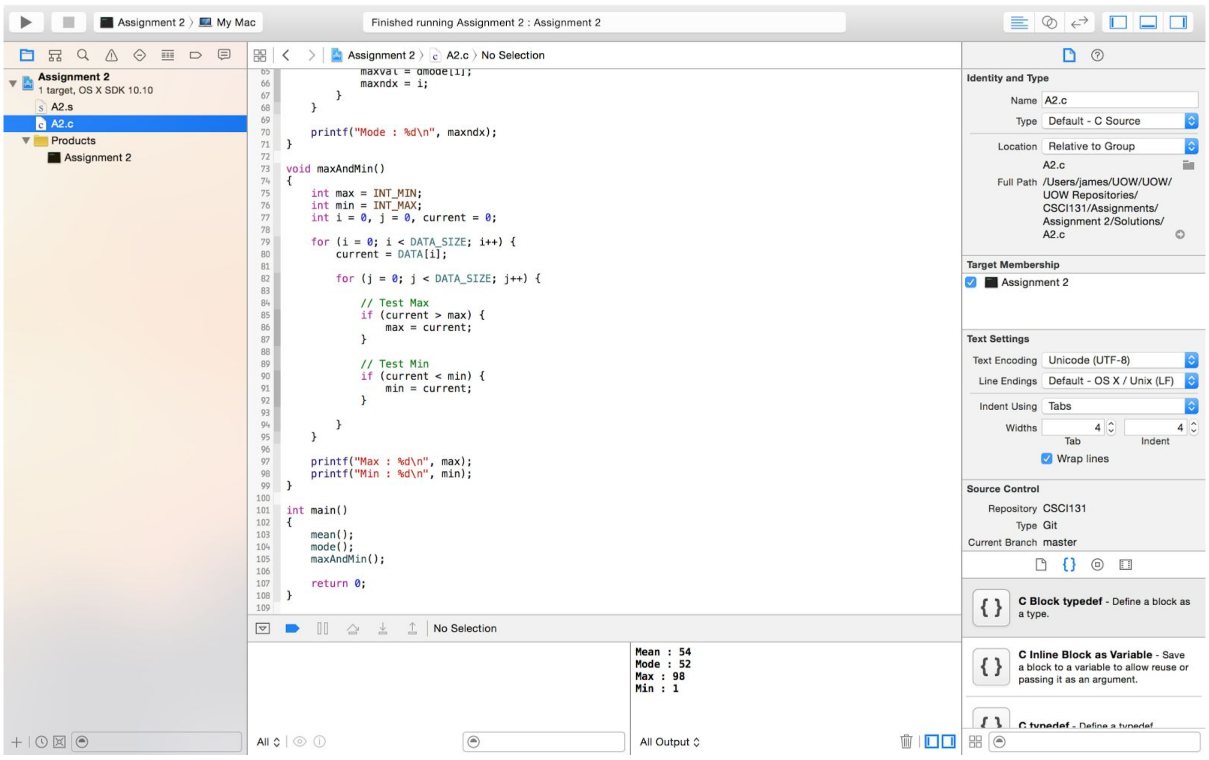
3.3 Results

C Code

Text Output:

```
Mean : 54
Mode : 52
Max : 98
Min : 1
```

Screenshot:



Finished running Assignment 2 : Assignment 2

Assignment 2 〉 A2.c 〉 No Selection

```c
                maxval = amode[i];
                maxndx = i;
            }
        }

    printf("Mode : %d\n", maxndx);
}

void maxAndMin()
{
    int max = INT_MIN;
    int min = INT_MAX;
    int i = 0, j = 0, current = 0;

    for (i = 0; i < DATA_SIZE; i++) {
        current = DATA[i];

        for (j = 0; j < DATA_SIZE; j++) {

            // Test Max
            if (current > max) {
                max = current;
            }

            // Test Min
            if (current < min) {
                min = current;
            }

        }
    }

    printf("Max : %d\n", max);
    printf("Min : %d\n", min);
}

int main()
{
    mean();
    mode();
    maxAndMin();

    return 0;
}
```

```
Mean : 54
Mode : 52
Max : 98
Min : 1
```

Identity and Type
Name: A2.c
Type: Default - C Source
Location: Relative to Group
A2.c
Full Path: /Users/james/UOW/UOW/UOW Repositories/CSCI131/Assignments/Assignment 2/Solutions/A2.c

Target Membership
Assignment 2

Text Settings
Text Encoding: Unicode (UTF-8)
Line Endings: Default - OS X / Unix (LF)
Indent Using: Tabs
Widths: 4 Tab, 4 Indent
Wrap lines

Source Control
Repository: CSCI131
Type: Git
Current Branch: master

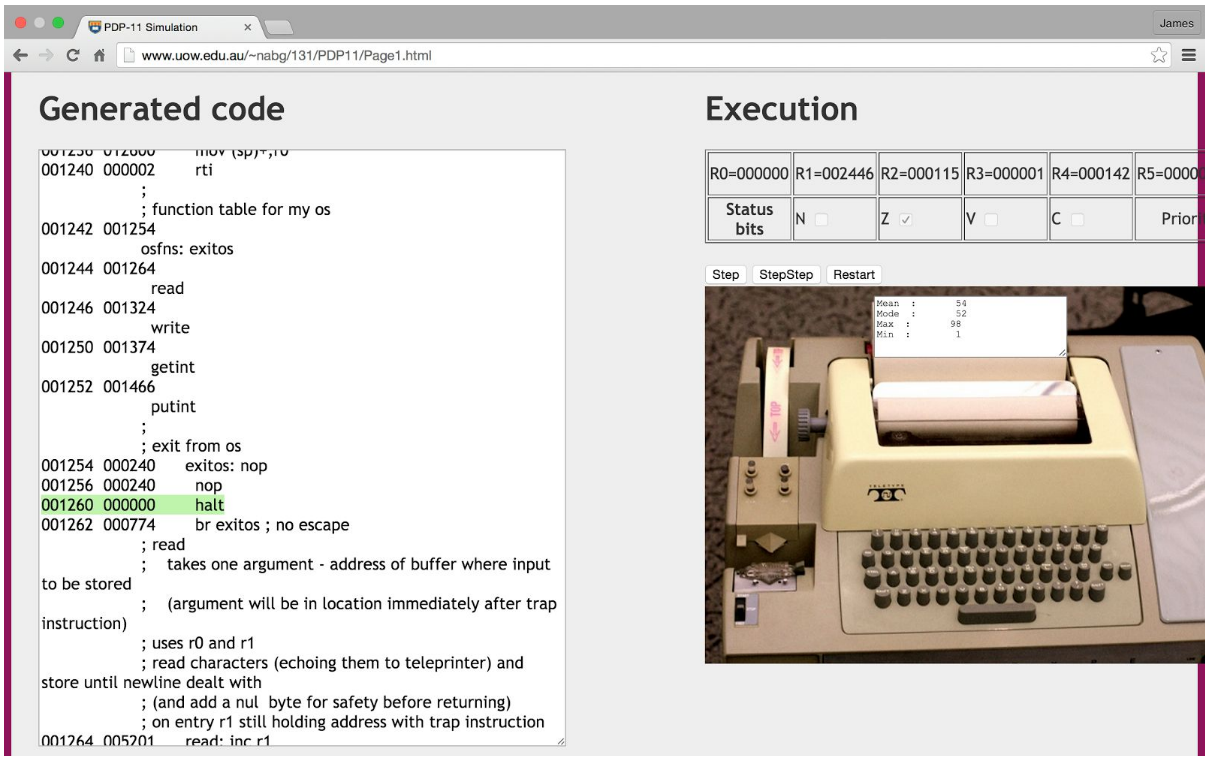
Assignment 2
My Mac
Assignment 2
1 target, OS X SDK 10.10
A2.s
A2.c
Products
Assignment 2
No Selection
All
All Output
C Block typedef - Define a block as a type.
C Inline Block as Variable - Save a block to a variable to allow reuse or passing it as an argument.
C typedef - Define a typedef

PDP-11 Assembly Code

Text Output

```
Mean :     54
Mode :     52
Max :     98
Min :      1
```

Screenshot:



PDP-11 Simulation

www.uow.edu.au/~nabg/131/PDP11/Page1.html

Generated code

```
001240  000002       rti
                  ;
                  ; function table for my os
001242  001254
                  osfns: exitos
001244  001264
                    read
001246  001324
                    write
001250  001374
                    getint
001252  001466
                    putint
                  ;
                  ; exit from os
001254  000240      exitos: nop
001256  000240       nop
001260  000000       halt
001262  000774       br exitos ; no escape
                  ; read
                  ;    takes one argument - address of buffer where input
to be stored
                  ;    (argument will be in location immediately after trap
instruction)
                  ; uses r0 and r1
                  ; read characters (echoing them to teleprinter) and
store until newline dealt with
                  ; (and add a nul  byte for safety before returning)
                  ; on entry r1 still holding address with trap instruction
```

Execution

R0=000000	R1=002446	R2=000115	R3=000001	R4=000142	R5=0000
Status bits	N	Z ✓	V	C	Prior

```
Mean  :       54
Mode  :       52
Max  :       98
Min  :        1
```

4. Conclusion

As we can see the program runs as expected and provides the appropriate output to calculations and answers as seen in the results above.