**EE2016 Report for Midterm**

**Name : Vishnu Varma V  
Roll No : EE19B059**

**OS :** Linux Ubuntu 18.04.5

**Procedure to Compile :**  
main.c - main C program which passes Array A and number of elements n as arguments

function\_q1.s - Code for question 1

function\_q2.s - Code for question 2

In terminal :

**For first question :**

$ Arm-linux-gnueabi-gcc -static -g -o question1 main.c function\_q1.s

$ ./question1

**For Second  question :**

$ Arm-linux-gnueabi-gcc -static -g -o question2 main.c function\_q2.s

$ ./question2

**Algorithm :**

We are given a sorted array of integers A[] and number of elements in the array

**Question 1:**

To access the elements of the array A[] we use ldr command and load A[0] to A[5] into registers r4-r9

We use this formula S[0] = A[5] - A[4] + A[3] - A[2] + A[1] - A[0]

To calculate S[0] in register r2

Loop variable (i) is initialised to 1 as S[1] must be compared to S[0] in the first iteration

The index of the minimum valued element in the array is in the register r12 initialised to 0

The register r1 is replaced with the value of (n-5) because in the loop (n-5) is used as S is defined for indices till (n-5) only

In the loop :

Each element of the array is shifted one register backward and the last register r9 is loaded with the next element A[i+5]

And using this formula : S[i] = A[i+5] - A[i+4] + A[i+3] - A[i+2] + A[i+1] - A[i]

Then S[i] is compared with the minimum value till then which was stored in r2

If minimum value till then > S[i] then minimum value becomes S[i]

And minimum index becomes i

Incrementing i by 1, comparing it with (n-5) and iterating the loop until i=(n-5)

Once the condition fails passing the minimum index value at r12 to r0 .

**Question 2:**  
To access the elements of the array A[] we use ldr command and load A[0] to A[5] into registers r4-r9

We use this formula S[0] = A[5] - A[4] + A[3] - A[2] + A[1] - A[0]

To calculate S[0] in register r2

The index of the minimum valued element in the array is in the register r12 initialised to 0

The register r1 is replaced with the value of (n-5) because in the loop (n-5) is used as S is defined for indices till (n-5) only

Loop variable (i) is initialised to 1 as S[1] must be compared to S[0] in the first iteration

Initialising the minimum value of S[] as S[0] and copying into r4

In Loop:

While accessing the array elements the address was incremented and points to A[i+6] to get the value at A[i] we decrease the address by 6\*4 = 24 and obtain the value using ldr command

Using the formula : S[i] = A[i+6] - A[i] -S[i-1]

Calculated S[i] and compared with the minimum value till then which was stored in r4

If minimum value till then > S[i] then minimum value becomes S[i]

And minimum index becomes i

Incrementing i by 1, comparing it with (n-5) and iterating the loop until i=(n-5)

Once the condition fails passing the minimum index value at r12 to r0 .

**ANSWER to question 1 :**

(a)Assuming load and store instructions cost 5 cycles each,

The number of cycles required by this code for a 100 element array is :

Str command for array A[] and n = 2\*5 = 10

Push 7 registers : 7\*5 =  35

6 ldr commands : 6\*5 = 30

8 other commands( sub,add,mov) before the loop = 8

In the loop :

5 mov commands = 5

1 ldr command = 1\*5 = 5

11 other commands = 11

End of the loop

1 mov command = 1

Pop 7 registers = 7\*5 = 35

Total number of clock cycles : 119 + (n-6)\*21

For 100 element array = 119 + (100-6)\*21 = 2093 clock cycles

(b) 7 registers are saved and restored in this code

**ANSWER to question 2 :**

1. Yes a faster algorithm is developed using this condition, the algorithm is mentioned above
2. Assuming load and store instructions cost 1 cycle each,

The number of cycles required by this code for a 100 element array is :

Str command for array A[] and n = 2\*1 = 2

Push 7 registers : 7\*1 =  7

6 ldr commands : 6\*1 = 6

9 other commands( sub,add,mov) before the loop = 9

In the loop :

2 ldr command = 2\*1 = 2

8 other commands = 8

End of the loop

1 mov command = 1

Pop 7 registers = 7\*1 = 7

Total number of clock cycles : 32 + (n-6)\*10

For 100 element array = 32+ (100-6)\*10 = 972 clock cycles

(c) 7 registers are saved and restored in this code

(d) This scheme couldn’t be used in 1st problem because the number of cycles taken by ldr was 5 whereas here it is 1, and number of ldr instructions was restricted to 1 in this case 2 ldr commands were used.

Number of total cycles for the case of LDR/STR instructions being x times as slow as other instructions :

Code 1 : (22\*x + 9) + (n-6)\*(16+x)

Code 2 : (22\*x + 10) + (n-6)\*(2\*x+8)

Table showing the number of clock cycles consumed by each code at different cost of ldr/str

n=100 for this table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Methods/ldr cycles | 1 cycle | 5 cycles | 10 cycles | 20 cycles |
| Code 1 | 1629 | 2093 | 2673 | 3833 |
| Code 2 | 972 | 1812 | 2862 | 4962 |

(e) If S[i] is expressed in terms of S[i- 1] then it speeds up the code, this method is used in the second code whereas in the first one the given formula in terms of A[i]’s is used.