

ARM Microcontroller Lab (23EEEP202)

Team -6
Srikrishna 459
Niharika 435
Karthik 413
Mahanthesh 406

Expt./ Job No.1: Write an ALP to achieve the following arithmetic operations:

i. Subtraction

```
data.asm
1  AREA AA, CODE, READONLY
2  ENTRY
3  LDR R0, =0x00000008; load the starting address
4  LDR R3, =0x00000002; load the second address
5  ADDS R2, R0, R3; add the data & store in reg r2
6  STOP B STOP
7
8  end
9
```

Output:

Register	Value	
R0	0x00000008	1 AREA AA, CODE, READONLY
R1	0x00000000	2 ENTRY
R2	0x0000000A	3 0.083 us LDR R0, =0x00000008; load the starting address
R3	0x00000002	4 0.083 us LDR R3, =0x00000002; load the second address
R4	0x00000000	5 0.083 us ADDS R2, R0, R3; add the data & store in reg r2
R5	0x00000000	6 0.250 us STOP B STOP
R6	0x00000000	7
R7	0x00000000	8 end
R8	0x00000000	9
R9	0x00000000	
R10	0x00000000	
R11	0x00000000	
R12	0x00000000	
R13 (SP)	0x00000000	
R14 (LR)	0x00000000	
R15 (PC)	0x0000000C	
CPSR	0x00000003	
SPSR	0x00000000	

ARM Microcontroller Lab (23EEEP202)

Alternate code:

```
data.asm
1  AREA AA, CODE, READONLY
2  ENTRY
3      LDR R0, =VAL1      ; Load address of first operand
4      LDR R1, =VAL2      ; Load address of second operand
5      LDR R2, [R0]        ; Load first value
6      LDR R3, [R1]        ; Load second value
7      sub R4, R2, R3      ; Perform addition using R4 as accumulator
8  STOP B STOP
9  VAL1 DCD 05
10 VAL2 DCD 01
11  END
12
```

Output:

Registers

Register	Value
Current	
R0	0x00000018
R1	0x0000001C
R2	0x00000005
R3	0x00000001
R4	0x00000004
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x000000D3
SPSR	0x00000000

data.asm

```

1  AREA AA, CODE, READONLY
2  ENTRY
3      LDR R0, =VAL1      ; Load address of first operand
4      LDR R1, =VAL2      ; Load address of second operand
5      LDR R2, [R0]        ; Load first value
6      LDR R3, [R1]        ; Load second value
7      sub R4, R2, R3      ; Perform addition using R4 as accumulator
8  STOP B STOP
9  VAL1 DCD 05
10 VAL2 DCD 01
11  END
12

```

ARM Microcontroller Lab (23EEEP202)

ii. 32 bit addition

```
data.asm*
1  AREA AA, CODE, READONLY
2  ENTRY
3  LDR R0, =0x00000008;load the starting address
4  LDR R3, =0x00000002 ;load the second address
5  ADDS R2, R0, R3 ;add the data & store in reg r2
6  STOP B STOP
7
8  end
9
```

Output:

Registers		data.asm	
Register	Value		
Current		1	AREA AA, CODE, READONLY
R0	0x00000008	2	ENTRY
R1	0x00000000	3	LDR R0, =0x00000008;load the starting address
R2	0x0000000A	4	LDR R3, =0x00000002 ;load the second address
R3	0x00000002	5	ADDS R2, R0, R3 ;add the data & store in reg r2
R4	0x00000000	6	STOP B STOP
R5	0x00000000	7	
R6	0x00000000	8	end
R7	0x00000000	9	
R8	0x00000000		
R9	0x00000000		
R10	0x00000000		
R11	0x00000000		
R12	0x00000000		
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x0000000C		
CPSR	0x000000D3		
SPSR	0x00000000		

Alternate code:



ARM Microcontroller Lab (23EEEP202)

```
data.asm
1  AREA AA, CODE, READONLY
2  ENTRY
3  LDR R0, =VAL1 ; Load address of first operand
4  LDR R1, =VAL2 ; Load address of second operand
5  LDR R2, [R0] ; Load first value
6  LDR R3, [R1] ; Load second value
7  ADD R4, R2, R3 ; Perform addition using R4 as accumulator
8  STOP B STOP
9  VAL1 DCD 03
10 VAL2 DCD 07
11 END
12
```

Output:

Registers		data.asm	
Register	Value		
Current		1	AREA AA, CODE, READONLY
R0	0x00000018	2	ENTRY
R1	0x0000001C	3	0.250 us LDR R0, =VAL1 ; Load address of first operand
R2	0x00000003	4	0.250 us LDR R1, =VAL2 ; Load address of second operand
R3	0x00000007	5	0.250 us LDR R2, [R0] ; Load first value
R4	0x0000000A	6	0.250 us LDR R3, [R1] ; Load second value
R5	0x00000000	7	0.083 us ADD R4, R2, R3 ; Perform addition using R4 as accumulator
R6	0x00000000	8	0.250 us STOP B STOP
R7	0x00000000	9	VAL1 DCD 03
R8	0x00000000	10	VAL2 DCD 07
R9	0x00000000	11	END 07 = 0x00000007
R10	0x00000000	12	
R11	0x00000000		
R12	0x00000000		
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x00000014		
CPSR	0x000000D3		
SPSR	0x00000000		

ARM Microcontroller Lab (23EEEP202)

iii. 64 bit addition

```

1  AREA AA, CODE, READONLY
2  ENTRY
3  LDR R0, =0X50      ;load 32-bit data
4  LDR R1, =0X90      ;load 32-bit data
5  ADDS R2, R0, R1     ;add 32-bit data
6  LDR R3, =0X70      ;load 32-bit data
7  LDR R4, =0X90      ;load 32-bit data
8  adds R5, R3, R4     ;add 32-bit data along with carry
9  STOP B STOP
10 END
11

```

Output:

Register	Value		
Current			
R0	0x00000050	1	AREA AA, CODE, READONLY
R1	0x00000090	2	ENTRY
R2	0x000000E0	3	0.083 us LDR R0, =0X50 ;load 32-bit data
R3	0x00000070	4	0.083 us LDR R1, =0X90 ;load 32-bit data
R4	0x00000090	5	0.083 us ADDS R2, R0, R1 ;add 32-bit data
R5	0x00000100	6	0.083 us LDR R3, =0X70 ;load 32-bit data
R6	0x00000000	7	0.083 us LDR R4, =0X90 ;load 32-bit data
R7	0x00000000	8	0.083 us adds R5, R3, R4 ;add 32-bit data along with carry
R8	0x00000000	9	0.250 us STOP B STOP
R9	0x00000000	10	END
R10	0x00000000	11	
R11	0x00000000		
R12	0x00000000		
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x00000018		
CPSR	0x000000D3		
SPSR	0x00000000		

Alternate code:

```

1  AREA CC, CODE, READONLY
2  ENTRY
3  LDMIA R0!, {R2, R3} ; Load 64-bit value
4  LDMIA R1!, {R4, R5} ; Load another 64-bit value
5  ADDS R6, R2, R4     ; Add lower 32-bits
6  ADC R7, R3, R5     ; Add upper 32-bits with carry
7  STOP B STOP
8
9  VAL1 DCD 0X50, 0X20
10 VAL2 DCD 0X40, 0X10
11 END
12

```

ARM Microcontroller Lab (23EEEP202)

Output:

Registers		data.asm	
Register	Value		
Current		1	AREA CC, CODE, READONLY
R0	0x00000008	2	ENTRY
R1	0x00000008	3	0.333 us LDMIA R0!, {R2, R3} ; Load 64-bit value
R2	0xE8B0000C	4	0.333 us LDMIA R1!, {R4, R5} ; Load another 64-bit value
R3	0xE8B10030	5	0.083 us ADDS R6, R2, R4 ; Add lower 32-bits
R4	0xE8B0000C	6	0.083 us ADC R7, R3, R5 ; Add upper 32-bits with carry
R5	0xE8B10030	7	0.250 us STOP B STOP
R6	0xD1600018	8	
R7	0xD1620061	9	VAL1 DCD 0X50, 0X20
R8	0x00000000	10	VAL2 DCD 0X60, 0X10
R9	0x00000000	11	END
R10	0x00000000	12	
R11	0x00000000		
R12	0x00000000		
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x00000010		
CPSR	0xA0000003		
SPSR	0x00000000		

ARM Microcontroller Lab (23EEEP202)

iv. Multiplication

```
data.asm
1 AREA AA, CODE, READONLY
2 ENTRY
3 LDR R0, =0X10      ;load 32-bit data
4 LDR R1, =0X40      ;load 32-bit data
5 muls R5, R1, R0     ;add 32-bit data along with carry
6 STOP B STOP
7 END
```

Output:

Register	Value
Current	
R0	0x00000010
R1	0x00000040
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000040
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x0000000C
CPSR	0x00000003
SPSR	0x00000000

1	AREA AA, CODE, READONLY
2	ENTRY
3	0.083 us LDR R0, =0X10 ;load 32-bit data
4	0.083 us LDR R1, =0X40 ;load 32-bit data
5	0.167 us muls R5, R1, R0 ;add 32-bit data along with carry
6	0.500 us STOP B STOP
7	END

ARM Microcontroller Lab (23EEEP202)

Alternate code:

```
data.asm
1
2 AREA AA, CODE, READONLY
3 ENTRY
4     LDR R0, =VAL1
5     LDR R1, =VAL2
6     LDR R2, [R0]
7     LDR R3, [R1]
8     muls R5, R3, R2      ;add 32-bit data along with carry
9 STOP B STOP
10 VAL1 DCD 0X20
11 VAL2 DCD 0X04
12
13 END
```

Output:

Registers

Register	Value
R0	0x00000018
R1	0x0000001C
R2	0x00000020
R3	0x00000004
R4	0x00000000
R5	0x00000080
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000014
CPSR	0x000000D3
SPSR	0x00000000

data.asm

```

1
2 AREA AA, CODE, READONLY
3 ENTRY
4     LDR R0, =VAL1
5     LDR R1, =VAL2
6     LDR R2, [R0]
7     LDR R3, [R1]
8     muls R5, R3, R2      ;add 32-bit data along with carry
9 STOP B STOP
10 VAL1 DCD 0X20
11 VAL2 DCD 0X04
12
13 END

```


ARM Microcontroller Lab (23EEEP202)

v. 32 bit binary divide

```
data.asm
1 AREA AA, CODE, READONLY
2 ENTRY
3 LDR R0, =0X10 ;load divisor into r0
4 LDR R1, =0X02 ;load dividend into r1
5 BACK SUBS R2, R0, R1
6 ADD R3, R3, #1
7 CMP R1, R2
8 BHI STOP
9 MOV R0, R2
10 b BACK
11 STOP B STOP
12 END
13
```

Output:

Register	Value		data.asm
Current			
R0	0x00000002		1 AREA AA, CODE, READONLY
R1	0x00000002		2 ENTRY
R2	0x00000000	0.083 us	3 LDR R0, =0X10 ;load divisor into r0
R3	0x00000008	0.083 us	4 LDR R1, =0X02 ;load dividend into r1
R4	0x00000000	0.667 us	5 BACK SUBS R2, R0, R1
R5	0x00000000	0.667 us	6 ADD R3, R3, #1
R6	0x00000000	0.667 us	7 CMP R1, R2
R7	0x00000000	0.833 us	8 BHI STOP
R8	0x00000000	0.583 us	9 MOV R0, R2
R9	0x00000000	1.750 us	10 b BACK
R10	0x00000000	1.250 us	11 STOP B STOP
R11	0x00000000		12 END
R12	0x00000000		13
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x00000020		
CPSR	0x20000003		
SPSR	0x00000000		

ARM Microcontroller Lab (23EEEP202)

Alternate code:

```
1 AREA AA, CODE, READONLY
2 ENTRY
3 LDR R0, =0x08 ;load divisor into r0
4 LDR R1, =0x04 ;load dividend into r1
5 BACK SUBS R2, R0, R1
6 ADD R3, R3, #1
7 CMP R1, R2
8 BHI STOP
9 MOV R0, R2
10 b BACK
11 STOP B STOP
12 END
13
```

Output:

Registers		data.asm	
Register	Value		
Current		1	AREA AA, CODE, READONLY
R0	0x00000004	2	ENTRY
R1	0x00000004	3	0.083 us LDR R0, =0x08 ;load divisor into r0
R2	0x00000000	4	0.083 us LDR R1, =0x04 ;load dividend into r1
R3	0x00000002	5	0.167 us BACK SUBS R2, R0, R1
R4	0x00000000	6	0.167 us ADD R3, R3, #1
R5	0x00000000	7	0.167 us CMP R1, R2
R6	0x00000000	8	0.083 us BHI STOP
R7	0x00000000	9	0.083 us MOV R0, R2
R8	0x00000000	10	0.250 us b BACK
R9	0x00000000	11	STOP B STOP
R10	0x00000000	12	END
R11	0x00000000	13	
R12	0x00000000		
R13 (SP)	0x00000000		
R14 (LR)	0x00000000		
R15 (PC)	0x00000014		
CPSR	0x20000003		
SPSR	0x00000000		

Expt./ Job No.2: Write an ALP for the following using loops:

ARM Microcontroller Lab (23EEEP202)

i. Find the sum of 'N' 16 bit numbers

```

1  AREA SUM, CODE, READONLY
2  ENTRY
3      LDR R0, =VAL1 ;initialise the sum value to zero
4      LDR R1, =0X00 ;initialize the counter
5      LDR R2, =0X05
6  LOOP LDRH R3, [R0], #0x02 ;take elements pointed by r0
7      ADD R1, R1, R3 ;add the numbers
8      SUBS R2, R2, #0X01 ;decrement the counter
9      BNE LOOP
10 STOP B STOP
11 VAL1 DCW 1,2,3,4,5,6
12
13 end
    
```

Output:

Register	Value	
R0	0x0000002A	1 AREA SUM, CODE, READONLY
R1	0x0000000F	2 ENTRY
R2	0x00000000	3 LDR R0, =VAL1 ;initialise the sum value to zero
R3	0x00000005	4 LDR R1, =0X00 ;initialize the counter
R4	0x00000000	5 LDR R2, =0X05
R5	0x00000000	6 LOOP LDRH R3, [R0], #0x02 ;take elements pointed by r0
R6	0x00000000	7 ADD R1, R1, R3 ;add the numbers
R7	0x00000000	8 SUBS R2, R2, #0X01 ;decrement the counter
R8	0x00000000	9 BNE LOOP
R9	0x00000000	10 STOP B STOP
R10	0x00000000	11 VAL1 DCW 1,2,3,4,5,6
R11	0x00000000	12
R12	0x00000000	13 end
R13 (SP)	0x00000000	
R14 (LR)	0x00000000	
R15 (PC)	0x0000001C	
CPSR	0x60000003	
SPSR	0x00000000	



KLE Technological
University
Creating Value
Leveraging Knowledge

KLE Technological University
School of Electronics and Communication Engineering

ARM Microcontroller Lab (23EEEP202)

Alternate code:

ARM Microcontroller Lab (23EEEP202)

ii. Find the maximum/minimum of N numbers

A.Maximum:

```
data.asm
1  area largest , code, readonly
2  entry
3  mov r5,#6          ; initialise counter to 6(i.e. n=7)
4  ldr r1,value1       ; loads the address of first value
5  ldr r2,[r1],#4      ; word align to 0 array element
6  loop
7  ldr r4,[r1],#4      ; word align to 0 array element
8  cmp r2,r4          ; compare numbers
9  bhi loop1          ; if the first number is > then goto loop1
10
11 mov r2,r4          ; if the first number is < then mov content r4 to r2
12 loop1
13 subs r5,r5,#1      ; decrement counter
14 cmp r5,#0          ; compare counter to 0
15 bne loop           ; loop back till array ends
16
17 ldr r4,result       ; loads the address of result
18 str r2,[r4]         ; stores the result in r2
19 stop b stop
20
21 ; array of 32 bit numbers(n=7)
22
23 value1
24 dcd 0x00000001      ;
25 dcd 0x000000020     ;
26 dcd 0x00000003      ;
27 dcd 0x000000089     ;
28 dcd 0x000000078     ;
29 dcd 0x00000006      ;
30 dcd 0x000000098     ;
31 area data2,data,readonly ; to store result in given address
32 result dcd 0x0
33 end
```

Output:

Register	Value				
Current					
R0	0x00000000				
R1	0x00000050				
R2	0x00000098				
R3	0x00000000				
R4	0x00000000				
R5	0x00000000				
R6	0x00000000				
R7	0x00000000				
R8	0x00000000				
R9	0x00000000				
R10	0x00000000				
R11	0x00000000				
R12	0x00000000				
R13 (SP)	0x00000000				
R14 (LR)	0x00000000				
R15 (PC)	0x00000030				
CPSR	0x600000D3				
SPSR	0x00000000				

Address	Disassembly	Comment
1	area largest , code, readonly	
2	entry	
3	mov r5,#6	; initialise counter to 6(i.e. n=7)
4	ldr r1,value1	; loads the address of first value
5	ldr r2,[r1],#4	; word align to 0 array element
6	loop	
7	ldr r4,[r1],#4	; word align to 0 array element
8	cmp r2,r4	; compare numbers
9	bhi loop1	; if the first number is > then goto loop1
10		
11	mov r2,r4	; if the first number is < then mov content r4 to r2
12	loop1	
13	subs r5,r5,#1	; decrement counter
14	cmp r5,#0	; compare counter to 0
15	bne loop	; loop back till array ends
16		
17	ldr r4,result	; loads the address of result
18	str r2,[r4]	; stores the result in r2
19	stop b stop	
20		
21		; array of 32 bit numbers(n=7)
22		
23	value1	

ARM Microcontroller Lab (23EEEP202)

Alternate code:

Register	Value
R0	0x00000000
R1	0x00000050
R2	0x00000009
R3	0x00000000
R4	0x40000000
R5	0x00000000
R6	0x00000000
R7	0x00000050
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000034
CPSR	0x600000D3
SPSR	0x00000000

```

1  area largest, code, readonly
2  entry
3  ldr r7, = value2
4  ldr r5, [r7]      ; initialise counter to 6 (i.e. n-7)
5  ldr r1, =value1    ; loads the address of first value
6  ldr r2, [r1], #4    ; word align to array element
7  loop
8  ldr r4, [r1], #4    ; word align to array element
9  cmp r2, r4         ; compare numbers
10 bhi loop1          ; if the first number is > then goto loop1
11
12 mov r2, r4         ; if the first number is < then mov content r4 to r2
13 loop1
14 sub r5, r5, #1     ; decrement counter
15 cmp r5, #0        ; compare counter to 0
16 bne loop          ; loop back till array ends
17
18 ldr r4, =result     ; loads the address of result
19 str r2, [r4]       ; stores the result in r2
20 stop b stop
21
22 ; array of 32 bit numbers (n=7)
23
24 value1
25 dcd 0x00000001
26 dcd 0x00000002
27 dcd 0x00000004
28 dcd 0x00000008
29 dcd 0x00000008
30 dcd 0x00000009
31
32 value2 dcd 0x06
33 area data2, data, readwrite ; to store result in given address
34 result dcd 0x00000000
35 end

```

ARM Microcontroller Lab (23EEEP202)

B. MINIMUM:

```

1  area smallest , code, readonly
2  mov r5,#6           ; initialise counter to 6(i.e. n=7)
3  ldr r1,value1        ; loads the address of first value
4  ldr r2,[r1],#4       ; word align to array element
5  loop
6  ldr r4,[r1],#4       ; word align to array element
7  cmp r2,r4           ; compare numbers
8  bls loop1           ; if the first number is < then goto loop1
9
10 mov r2,r4            ; if the first number is > then mov content r4 to r2
11 loop1
12 subs r5,r5,#1        ; decrement counter
13 cmp r5,#0           ; compare counter to 0
14 bne loop            ; loop back till array ends
15
16 ldr r4,result        ; loads the address of result
17 str r2,[r4]          ; stores the result in r1
18 stop b stop
19 ; array of 32 bit numbers(n=7)
20
21 value1
22 dcd 0x00000001
23 dcd 0x00000005
24 dcd 0x00000090
25 dcd 0x00000004
26 dcd 0x00000079
27 dcd 0x00000006
28 dcd 0x0000003d
29
30 area data2,data,readonly ; to store result in given address
31 result dcd 0x0
32
33 end

```

Output:

Register	Value				
Current					
R0	0x00000000		1	0.083 us	area smallest , code, readonly
R1	0x00000050		2	0.250 us	mov r5,#6 ; initialise counter to 6(i.e. n=7)
R2	0x00000001		3	0.250 us	ldr r1,value1 ; loads the address of first value
R3	0x00000000		4	0.250 us	ldr r2,[r1],#4 ; word align to array element
R4	0x00000000		5		loop
R5	0x00000000		6	1.500 us	ldr r4,[r1],#4 ; word align to array element
R6	0x00000000		7	0.500 us	cmp r2,r4 ; compare numbers
R7	0x00000000		8	1.500 us	bls loop1 ; if the first number is < then goto loop1
R8	0x00000000		9		
R9	0x00000000		10		mov r2,r4 ; if the first number is > then mov content r4 to r2
R10	0x00000000		11		loop1
R11	0x00000000		12	0.500 us	subs r5,r5,#1 ; decrement counter
R12	0x00000000		13	0.500 us	cmp r5,#0 ; compare counter to 0
R13	0x00000000		14	1.333 us	bne loop ; loop back till array ends
R14	0x00000000		15		
R15	0x00000030		16	0.250 us	ldr r4,result ; loads the address of result
CPSR	0x600000D3		17	0.167 us	str r2,[r4] ; stores the result in r1
SPSR	0x00000000		18	0.750 us	stop b stop
			19		; array of 32 bit numbers(n=7)
			20		
			21		value1
			22		dcd 0x00000001 ;
			23		dcd 0x00000005 ;

ARM Microcontroller Lab (23EEEP202)

Alternate code:

```

1  area smallest , code, readonly
2  | entry
3      ldr r7,=value2
4      ldr r5,[r7]          ; initialise counter to 6(i.e. n=7)
5      ldr r1,=value1       ; loads the address of first value
6      ldr r2,[r1],#4       ; word align to array element
7  loop
8      ldr r4,[r1],#4       ; word align to array element
9      cmp r2,r4           ; compare numbers
10     bls loop1            ; if the first number is < then goto loop1
11
12     mov r2,r4            ; if the first number is > then mov content r4 to r2
13 loop1
14     subs r5,r5,#1        ; decrement counter
15     cmp r5,#0           ; compare counter to 0
16     bne loop            ; loop back till array ends
17
18     ldr r4,=result       ; loads the address of result
19     str r2,[r4]          ; stores the result in r1
20 stop b stop
21 ; array of 32 bit numbers(n=7)
22
23 value1
24     dcd 0x00000001
25     dcd 0x00000000
26     dcd 0x00000003
27     dcd 0x00000004
28     dcd 0x00000009
29     dcd 0x00000006
30     dcd 0x00000007
31 value2 dcd 0x06
32 area data2,data,readonly ; to store result in given address
33 result dcd 0x0

```

```

1  area smallest , code, readonly
2  | entry
3      ldr r7,=value2
4      ldr r5,[r7]          ; initialise counter to 6(i.e. n=7)
5      ldr r1,=value1       ; loads the address of first value
6      ldr r2,[r1],#4       ; word align to array element
7  loop
8      ldr r4,[r1],#4       ; word align to array element
9      cmp r2,r4           ; compare numbers
10     bls loop1            ; if the first number is < then goto loop1
11
12     mov r2,r4            ; if the first number is > then mov content r4 to r2
13 loop1
14     subs r5,r5,#1        ; decrement counter
15     cmp r5,#0           ; compare counter to 0
16     bne loop            ; loop back till array ends
17
18     ldr r4,=result       ; loads the address of result
19     str r2,[r4]          ; stores the result in r1
20 stop b stop
21 ; array of 32 bit numbers(n=7)
22
23 value1
24     dcd 0x00000001
25     dcd 0x00000000
26     dcd 0x00000003
27     dcd 0x00000004
28     dcd 0x00000009
29     dcd 0x00000006
30     dcd 0x00000007
31 value2 dcd 0x06
32 area data2,data,readonly ; to store result in given address
33 result dcd 0x0
34
35
36 end

```


ARM Microcontroller Lab (23EEEP202)

- ii. Find the factorial of a given number with and without a look up table.
Apply suitable machine dependent optimization technique and analyze for memory and time consumed.

A. Write an ALP to find factorial of a given number with lookup table:

```

1  TTL FACTORIAL
2  AREA FACT, CODE, READONLY
3  ENTRY
4
5  MAIN
6      LDR R0, =DATA1
7      LDR R1, =VAL
8      LDR R1, [R1]
9      MOV R1, R1, LSL #2
10     ADD R0, R0, R1
11     LDR R2, [R0]
12     LDR R3, =RESULT
13     STR R2, [R3]
14
15 STOP B STOP
16
17     AREA DATA1, DATA, ALIGN=4
18 DATA1
19     DCD 1, 2, 6, 24, 120, 720, 5040
20 VAL
21     DCD 5
22 RESULT
23     DCD 0
24
25     END
26

```

Output:

r0	00000044	
r1	00000014	
r2	000002d0	
r3	00000050	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	00000020	
cpsr	000001d3	NZCVI SVC
spsr	00000000	NZCVI ?

ARM Microcontroller Lab (23EEEP202)

Alternate code:

```

1  TTL FACTORIAL LOOKUP
2  AREA FACT_ALT, CODE, READONLY
3  ENTRY
4
5  MAIN
6      LDR R1, =VAL
7      LDR R1, [R1]
8      LDR R0, =DATA1
9      LDR R2, [R0, R1, LSL #2]
10     LDR R3, =RESULT
11     STR R2, [R3]
12
13 STOP B STOP
14
15     AREA DATA1, DATA, ALIGN=4
16 DATA1
17     DCD 1, 2, 6, 24, 120, 720, 5040
18 VAL
19     DCD 5
20 RESULT
21     DCD 0
22
23     END
24

```

Output:

r0	00000044	
r1	00000014	
r2	000002d0	
r3	00000050	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	00000020	
cpsr	000001d3	NZCVI SVC
spsr	00000000	NZCVI ?

B. Write an ALP to find factorial of a given number without lookup table:



ARM Microcontroller Lab (23EEEP202)

```
withoutoup.s
1  TTL FACTORIAL
2  AREA PGM, CODE, READONLY
3  ENTRY
4  MAIN
5      LDR R0, =NUM
6      MOV R1, #1
7      MOV R2, #1
8
9  LOOP
10     MUL R3, R2, R1
11     MOV R1, R3
12     CMP R2, R0
13     BGT ENDFACTOR
14     ADD R2, R2, #1
15     B LOOP
16
17 STOP
18     B STOP
19     END
20
21 AREA FACTORIAL, CODE, READONLY
22 ENTRY
23 MAIN
24     MOV R0, #10
25     MOV R1, R0
26     MOV R3, R0
27
28 LOOP
29     SUBS R1, R1, #1
30     MULNE R2, R1, R0
31     MOV R0, R2
32     BNE LOOP
33
34 STOP
35     B STOP
36     END
```

Output:

r0	00000005	
r1	000002d0	
r2	00000006	
r3	000002d0	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	00000024	
cpsr	200001d3	NZCVI SVC
spsr	00000000	NZCVI ?

s0	00000000
s1	00000000
s2	00000000
s3	00000000



ARM Microcontroller Lab (23EEEP202)

Alternate code:

```
withoutloop.s
1      TTL FACTORIAL
2      AREA PGM, CODE, READONLY
3      ENTRY
4
5 MAIN
6      MOV R0, #5
7      MOV R1, R0
8      MOV R2, #1
9
10     LOOP
11     MUL R2, R2, R1
12     SUBS R1, R1, #1
13     BNE LOOP
14
15 STOP
16     B STOP
17     END
18
19     AREA FACTORIAL, CODE, READONLY
20     ENTRY
21
22 MAIN
23     MOV R0, #10
24     MOV R1, R0
25     MOV R2, #1
26
27     LOOP
28     MUL R2, R2, R1
29     SUBS R1, R1, #1
30     BNE LOOP
31
32 STOP
33     B STOP
34     END
35
```

Output:

r0	00000005	
r1	00000000	
r2	00000078	
r3	00000000	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	00000018	
cpsr	600001d3	NZCVI SVC
spsr	00000000	NZCVI ?

ARM Microcontroller Lab (23EEEP202)

Expt./ Job No.3: Write an ALP to:

i. Find the length of the carriage return terminated string

```
length.s*
1      TTL LENGTH
2      cr EQU 0x0D
3      AREA stringlength, CODE, READONLY
4      ENTRY
5
6 main
7      LDR R2, =0x00
8      LDR R0, =array
9
10 up
11     LDRB R1, [R0], #1
12     CMP R1, #cr
13     BEQ stop
14     ADD R2, R2, #1
15     BAL up
16
17 stop
18     B stop
19
20     AREA dd, CODE, READONLY
21 array DCB "hello world", cr
22     END
23
24
```

Output:

r0	00000034	
r1	0000000d	
r2	0000000b	
r3	00000000	
r4	00000000	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	0000001c	
cpsr	600001d3	NZCVI SVC
spsr	00000000	NZCVI ?

Alternate code:

ARM Microcontroller Lab (23EEEP202)

```

length.s
1      TTL LENGTH
2      cr EQU 0x0D
3      AREA stringlength, CODE, READONLY
4      ENTRY
5
6 main
7      MOV R2, #0          ; Initialize length counter
8      LDR R0, =array      ; Load address of string
9
10 up
11     LDRB R1, [R0], #1    ; Load byte and increment address
12     ADDNE R2, R2, #1     ; Increment length if not 0x0D
13     CMP R1, #cr         ; Compare with carriage return (0x0D)
14     BNE up              ; Loop if not found
15
16 stop
17     B stop              ; Infinite loop to halt
18
19     AREA dd, CODE, READONLY
20 array DCB "hello world", cr
21     END
22
23
24

```

Output:

r0	00000034		
r1	0000000d		
r2	0000000b		
r3	00000000		
r4	00000000		
r5	00000000		
r6	00000000		
r7	00000000		
r8	00000000		
r9	00000000		
r10	00000000		
r11	00000000		
r12	00000000		
sp	00000000		
lr	00000000		
pc	0000001c		
cpsr	600001d3	NZCVI	SVC
spsr	00000000	NZCVI	?

ii. Write an ALP to compare two strings for equality

ARM Microcontroller Lab (23EEEP202)

```
scomp.s*
1      AREA stringcompare, CODE, READONLY
2      ENTRY
3
4 main
5      LDR R0, =array1
6      LDR R1, =array2
7      MOV R2, #0x6
8
9 up
10     LDRB R3, [R0], #1
11     LDRB R4, [R1], #1
12     CMP R3, R4
13     BNE unequal
14     SUBS R2, R2, #1
15     BNE up
16
17     MOV R5, #0xFF
18     B stop
19
20 unequal
21     MOV R5, #0x00
22
23 stop
24     B stop
25
26     AREA dd, CODE, READONLY
27 val DCD 0x2020
28 array1 DCB "hello", 0x00
29 array2 DCB "hellooo", 0x00
30     END
31
32
33
```

Output:

r0	0000004a	
r1	00000050	
r2	00000001	
r3	00000000	
r4	0000006f	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	00000030	
cpsr	800001d3	NZCVI SVC
spsr	00000000	NZCVI ?

Alternate code:

ARM Microcontroller Lab (23EEEP202)

```
scomp.s*
1      AREA stringcompare, CODE, READONLY
2      ENTRY
3
4 main
5      LDR R0, =array1
6      LDR R1, =array2
7      MOV R2, #6
8
9 compare_loop:
10     LDRB R3, [R0], #1
11     LDRB R4, [R1], #1
12     SUBS R2, R2, #1
13     CMP R3, R4
14     BNE not_equal
15     BGT compare_loop
16
17     MOV R5, #0xFF
18     B stop
19
20 not_equal:
21     MOV R5, #0x00
22
23 stop:
24     B stop
25
26     AREA dd, CODE, READONLY
27 val DCD 0x2020
28 array1 DCB "hello", 0x00
29 array2 DCB "hellooo", 0x00
30     END
31
32
33
```

Output:

r0	0000004a	
r1	00000050	
r2	00000001	
r3	00000000	
r4	0000006f	
r5	00000000	
r6	00000000	
r7	00000000	
r8	00000000	
r9	00000000	
r10	00000000	
r11	00000000	
r12	00000000	
sp	00000000	
lr	00000000	
pc	00000030	
cpsr	800001d3	NZCVI SVC
spsr	00000000	NZCVI ?

ARM Microcontroller Lab (23EEEP202)

4. Write an ALP to pass parameters to a subroutine to find the factorial of a number or prime number generation.

Apply suitable machine dependent optimization technique and analyze for memory and time consumed

```
checkp.s*
1      AREA primegen, CODE, READONLY
2      ENTRY
3
4 main
5      LDR R0, =value
6      LDR R6, =0x40000000
7      MOV R8, #0x0A
8
9 done
10     LDR R1, [R0], #4
11     LSR R9, R1, #1
12     MOV R5, R1
13     MOV R4, #2
14     BL subb
15
16 back
17     CMP R3, #1
18     BGE prime
19
20 goo
21     SUB R8, R8, #1
22     CMP R8, #0
23     BNE done
24     B stop
25
26 prime
27     STR R1, [R6], #4
28     SUB R8, R8, #1
29     CMP R8, #0
30     BNE done
31     B stop
32
33 subb
34     MOV R3, R1
35
36 loop
```



ARM Microcontroller Lab (23EEEP202)

```
checkp.s*
25
26 prime
27     STR R1, [R6], #4
28     SUB R8, R8, #1
29     CMP R8, #0
30     BNE done
31     B stop
32
33 subb
34     MOV R3, R1
35
36 loop
37     SUB R3, R3, R4
38     CMP R3, R4
39     BGE loop
40
41     CMP R3, #0
42     BEQ goo
43     CMP R4, R3
44     BEQ back
45
46     ADD R4, R4, #1
47     SUB R9, R9, #1
48     CMP R9, #0
49     BNE subb
50
51     MOV PC, LR
52
53 stop
54     B stop
55
56 value
57     DCD 0x00000009, 0x00000008, 0x00000010, 0x0000000B, 0x00000003, 0x000000B7, 0x00000033, 0x00000012, 0x00000038, 0x00000007
58     END
59
```

Output:

r0	0000008c		
r1	00000009		
r2	00000003		
r3	00000000		
r4	00000003		
r5	00000009		
r6	40000000	40000000	0000000b
r7	00000000		
r8	0000000a	40000004	00000003
r9	00000009		
r10	00000000	40000008	0000000b7
r11	00000000		
r12	00000000	4000000c	00000007
sp	00000000		
lr	0000001c		
pc	0000001c		
cpsr	600001d3	NZCVI SVC	
spsr	00000000	NZCVI ?	

Alternate code:

ARM Microcontroller Lab (23EEEP202)

```

checkp.s
1      AREA primegen, CODE, READONLY
2      ENTRY
3
4 main
5      LDR R0, =value
6      LDR R6, =0x40000000
7      MOV R8, #10
8
9 next
10     LDR R1, [R0], #4
11     MOV R4, #2
12     MOV R3, R1
13     BL check_prime
14
15     CMP R3, #1
16     BNE skip
17     STR R1, [R6], #4
18
19 skip
20     SUBS R8, R8, #1
21     BNE next
22     B stop
23
24 check_prime
25     CMP R1, #2
26     BEQ prime
27
28     LSR R9, R1, #1
29
30 loop
31     UDIV R5, R1, R4
32     MUL R5, R5, R4
33     CMP R5, R1
34     BEQ not_prime
35
36     ADD R4, R4, #1
37
38     CMP R4, R9
39     BLS loop
40 prime
41     MOV R3, #1
42     MOV PC, LR
43
44 not_prime
45     MOV R3, #0
46     MOV PC, LR
47
48 stop
49     B stop
50
51 value
52     DCD 0x00000009, 0x00000008, 0x00000010, 0x0000000B, 0x00000003, 0x000000B7, 0x00000033, 0x00000012, 0x00000038, 0x00000007
53     END
54

```

Output:

ARM Microcontroller Lab (23EEEP202)

r0	0000008c		
r1	00000009		
r2	00000003		
r3	00000000		
r4	00000003		
r5	00000009		
r6	40000000	40000000	0000000b
r7	00000000		
r8	0000000a	40000004	00000003
r9	00000009		
r10	00000000	40000008	000000b7
r11	00000000		
r12	00000000	4000000c	00000007
sp	00000000		
lr	0000001c		
pc	0000001c		
cpsr	600001d3	NZCVI SVC	
spsr	00000000	NZCVI ?	

MEMEORY AND TIME

EXPT-1

CODES	TIME	MEMEORY
CODE i-a (sub)	0.490 us	
i-b	1.333 us	
CODE ii-a (add)	0.490 us	
ii-b	1.333 us	
CODE iii-a(64bit-add)	0.748 us	
iii- b	1.082 us	
CODE iv-a(mul)	0.833 us	
iv- b	1.417 us	
CODE v-a(div)	5.75 us	
v- b	1.083 us	

EXPT-2

CODES	TIME	MEMEORY
CODE i-a('N'16bit-sum)	4.833 us	
i-b		
CODE ii-a(max)	8.833 us	
ii-b		
CODE iii-a(min)	7.583 us	
iii-b		