

6. KEIL MICROVISION IDE ASSEMBLY LANGUAGE PROGRAMMING

6.1. FAMILIARIZING THE KEIL MICROVISIONV IDE

6.1.1. CREATE A PROJECT, EDIT AN ASM FILE, BUILD, AND DEBUG. OBSERVE DISASSEMBLY WINDOW, REGISTER AND MEMORY CONTENTS IN STEP MODE AND IN RUN MODE.

6.1.2. EXECUTE A SAMPLE ARM ASSEMBLY LANGUAGE PROGRAM TO ADD TWO NUMBERS IN REGISTERS AND STORE THE SUM IN A REGISTER.

6.2. ARM ASSEMBLY LANGUAGE PROGRAMMING PRACTICE USING KEIL MICROVISIONV # I

6.2.1. ALP TO ADD FIRST 5 NATURAL NUMBERS. STORE SUM IN REGISTER.

6.2.2. ALP TO ADD FIRST 10 ODD NUMBERS. STORE SUM IN REGISTER.

6.2.3. ALP TO COMPUTE SUM OF 5 TERMS OF AN ARITHMETIC PROGRESSION. FIRST TERM IS 3, COMMON DIFFERENCE IS 7. STORE SUM IN REGISTER.

6.2.4. ALP TO COMPUTE SUM OF SQUARES OF 5 NUMBERS STARTING FROM 1. WRITE AND USE PROCEDURE SQU. STORE SUM IN REGISTER.

6.3. ARM ASSEMBLY LANGUAGE PROGRAMMING PRACTICE USING KEIL MICROVISIONV # II

6.3.1. ALP TO ADD THE FIRST N EVEN NUMBERS. STORE THE RESULT IN A MEMORY LOCATION.

6.3.2. ALP TO GENERATE A GEOMETRIC PROGRESSION WITH A LIMIT N. DISPLAY THE RESULTS IN MEMORY.

6.4. ARM ALP # I

6.4.1. ALP TO FIND THE ARITHMETIC PROGRESSION WITH $A=3$, $D=7$.

6.4.2. ALP TO FIND THE SUM OF CUBES OF THE FIRST N NATURAL NUMBERS.

6.5. ARM ALP # II

6.5.1. ALP TO COUNT THE NUMBER OF ZEROES AND ONES IN A BINARY NUMBER.

6.5.2. ALP TO FIND THE AVERAGE OF TEN 16-BIT NUMBERS STORED IN MEMORY.

6.6. ARM ALP # III

6.6.1. ALP TO FIND THE FACTORIAL OF A NUMBER.

6.6.2. ALP TO GENERATE THE FIRST N FIBONACCI NUMBERS.

6.7. ARM ALP # IV

6.7.1. ALP TO FIND THE SUM OF DIGITS OF A NUMBER.

EXERCISE # 6.1

6.1.2) EXECUTE A SAMPLE ARM ASSEMBLY LANGUAGE PROGRAM TO ADD TWO NUMBERS IN REGISTERS AND STORE THE SUM IN A REGISTER.


```
AREA PROG1, CODE, READONLY
ENTRY
MOV R0, #0x78
MOV R1, #0x21
ADD R3, R1, R0
STOP B STOP
END
```

✓ EXERCISE # 6.2 ARM ASSEMBLY LANGUAGE PROGRAMMING PRACTICE USING
 KEIL MICROVISIONV # 2 6.2.1). ALP TO ADD FIRST 5 NATURAL NUMBERS. STORE
 SUM IN REGISTER.

```
AREA PROG2, CODE, READONLY
ENTRY
MOV R0, #0
MOV R1, #0
BACKK ADD R0, R0, #1
ADD R1, R1, R0
CMP R0, #5
BNE BACKK
GO B GO
END
```

✓ 6.2.2) ALP TO ADD FIRST 10 ODD NUMBERS. STORE SUM IN REGISTER.

```
AREA PROG3, CODE, READONLY
ENTRY
MOV R1, #1
MOV R2, #9
MOV R3, #1
BACKK ADD R3, R3, #2
ADD R1, R1, R3
SUBS R2, R2, #1
BNE BACKK
GO B GO
END
```



6.2.3) ALP TO COMPUTE SUM OF 5 TERMS OF AN ARITHMETIC PROGRESSION.
FIRST TERM IS 3, COMMON DIFFERENCE IS 7. STORE SUM IN REGISTER.

AREA PROG4,CODE,READONLY

ENTRY

MOV R3,#0

MOV R1,#3

MOV R2,#0

BACKK ADD R3,R3,R1

ADD R1,R1,#7

ADD R2,R2,#1

CMP R2,#5

BNE BACKK

GO B GO

END

Imp



6.2.4) ALP TO COMPUTE SUM OF SQUARES OF 5 NUMBERS STARTING FROM 1.

WRITE AND USE PROCEDURE SQU. STORE SUM IN REGISTER.

AREA **PROG5**,CODE,READONLY

ENTRY

MOV R7,#0

MOV R2,#1

LOOP BL **SQU**

ADD R7,R7,R4

ADD R2,R2,#1

CMP R2,#6

BNE **LOOP**

GO B GO

SQU MUL R4,R2,R2

MOV PC,LR

END



EXERCISE # 6.3 ARM ASSEMBLY LANGUAGE PROGRAMMING PRACTICE USING KEIL MICROVISIONV # II

6.3.1) ALP TO ADD THE FIRST N EVEN NUMBERS. STORE THE RESULT IN A
MEMORY LOCATION.

```
AREA PROG6 CODE, READONLY
N RN 1
RESULT RN 2
EVEN_NUMBER RN 3
ENTRY
MOV N, #5
MOV RESULT, #0
MOV EVEN_NUMBER, #2
MOV R4, #0x40000000
LOOP ADD RESULT, RESULT, EVEN_NUMBER
ADD EVEN_NUMBER, EVEN_NUMBER, #2
SUBS N, N, #1
BNE LOOP
STR RESULT, [R4]
STOP B STOP
END
```

6.3.2) ALP TO GENERATE A GEOMETRIC PROGRESSION WITH A LIMIT N. DISPLAY THE RESULTS IN MEMORY.

AREA PROG7,CODE,READONLY

A RN 1

D RN 2

N RN 3

ENTRY

MOV A,#1

MOV D,#2

MOV N,#10

MOV R5,#0x40000000

LOOP MUL R6,A,D

MOV A,R6

STR A,[R5],#4

SUBS N,N,#1


BNE LOOP

STOP B STOP

END

EXERCISE # 6.4 ARM ALP # I

6.4.1) ALP TO FIND THE ARITHMETIC PROGRESSION WITH A=3, D=7.

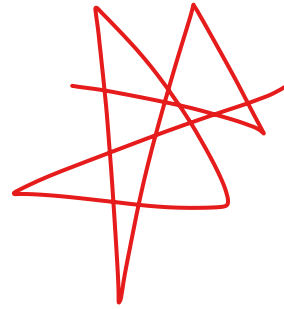


```
AREA PROG8,CODE,READONLY
ENTRY
MOV R1,#3
MOV R2,#1
LDR R3,=PRO
STR R1,[R3]
ADD R1,R1,#7
BACKK STR R1,[R3,#4]!
ADD R1,R1,#7
ADD R2,R2,#1
CMP R2,#10
BNE BACKK
GO B GO
AREA PROGRESSION,DATA,READWRITE
PRO SPACE 10
END
```



6.4.2) ALP TO FIND THE SUM OF CUBES OF THE FIRST N NATURAL NUMBERS.

```
AREA PROG9 CODE,READONLY
N RN 1
NPLUSONE RN 2
TEMP RN 3
RESULT RN 4
ENTRY
MOV R5,#0x40000000
LDR N,=3
ADD NPLUSONE,N,#1
MUL TEMP,N,NPLUSONE
MOV TEMP,TEMP,LSR #1
MUL RESULT,TEMP,TEMP
STR RESULT,[R5]
STOP B STOP
END
```



EXERCISE # 6.5 ARM ALP # II

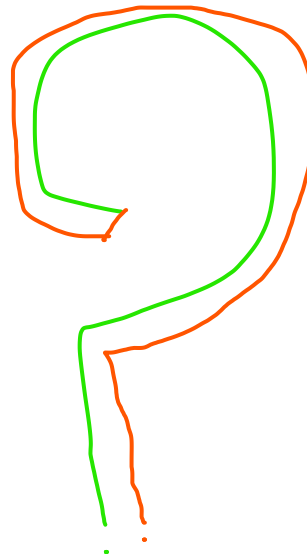
6.5.1) ALP TO COUNT THE NUMBER OF ZEROES AND ONES IN A BINARY NUMBER.

```
AREA PROG10, CODE, READONLY
NUMBER RN 1
NUMONES RN 10
NUMZEROES RN 11
ENTRY
MOV R5, #0x40000000
LDR NUMBER, =0xA
MOV NUMONES, #0
MOV NUMZEROES, #0
LOOP LSRs NUMBER, #1
ADDCS NUMONES, #1
ADDCC NUMZEROES, #1
CMP NUMBER, #0
BNE LOOP
STR NUMONES, [R5]
STR NUMZEROES, [R5, #4]
STOP B STOP
END
```

6.5.2) ALP TO FIND THE AVERAGE OF TEN 16-BIT NUMBERS STORED IN MEMORY.

```
AREA PROG11, CODE, READONLY
ENTRY
LDR R7, =TABLE
MOV R0, #9
LDRH R1, [R7]
BACKK LDRH R2, [R7, #2]!
ADD R1, R1, R2
SUBS R0, R0, #1
BNE BACKK
MOV R3, #10
MOV R4, #0
MOV R5, R1
BACKK1 SUBS R5, R5, R3
ADDPL R4, R4, #1
BPL BACKK1

ADDMI R5, R5, R3
GO B GO
TABLE DCW 1000, 2564, 8936, 344, 5667, 908, 786, 654, 9871, 456
END
```



EXERCISE # 6.6 ARM ALP # III

6.6.1) ALP TO FIND THE FACTORIAL OF A NUMBER.

AREA PROG12,CODE,READONLY

N RN 1

FACT RN 2

ENTRY

MOV N,#10

MOV FACT,#1

LOOP MUL FACT,N,FACT

SUBS N,N,#1

BNE LOOP

STOP B STOP

END

Imp

6.6.2) ALP TO GENERATE THE FIRST N FIBONACCI NUMBERS.

```

AREA PROG13, CODE, READONLY
ENTRY
MOV R1, #1
LDR R2, =TABLE
LDR R3, =NUMFIBONACCI
LDRB R6, [R3]
STRB R1, [R2], #1
MOV R3, #0
MOV R4, #0
MOV R5, #1
SUB R6, R6, #1
BACKK ADD R4, R3, R1
STRB R4, [R2], #1
MOV R3, R1
MOV R1, R4
ADD R5, R5, #1
CMP R5, R6
BLS BACKK
GO B GO

NUMFIBONACCI DCB 0x0A
AREA NUMBER, DATA, READWRITE
TABLE SPACE 60
END

```

EXERCISE # 6.7 ARM ALP # IV

6.7.1) ALP TO FIND THE SUM OF DIGITS OF A NUMBER.

```

AREA PROG14, CODE, READONLY
DIVIDEND RN 1
DIVISOR RN 2
QUOTIENT RN 3
REMAINDER RN 4
RESULT RN 5
ENTRY
LDR DIVIDEND, =12345
MOV DIVISOR, #10
MOV RESULT, #0
LOOP BL DIV
ADD RESULT, REMAINDER, RESULT
CMP QUOTIENT, #0
MOVNE DIVIDEND, QUOTIENT
BNE LOOP
STOP B STOP
DIV MOV QUOTIENT, #0
LOOP2 SUBS DIVIDEND, DIVIDEND, DIVISOR
ADDPL QUOTIENT, QUOTIENT, #1; QUOTIENT
BPL LOOP2
ADDMI REMAINDER, DIVIDEND, DIVISOR
BX LR
END

```