

# Advanced Computer Organization and Architecture with Lab

## Lab Assignment 5 – Code conversion

Submitted by – Sanjana Jammi ( CS18M522)

### Screenshots:

**Part 1** - Convert an ASCII digit(given in hex format) to hex digit.

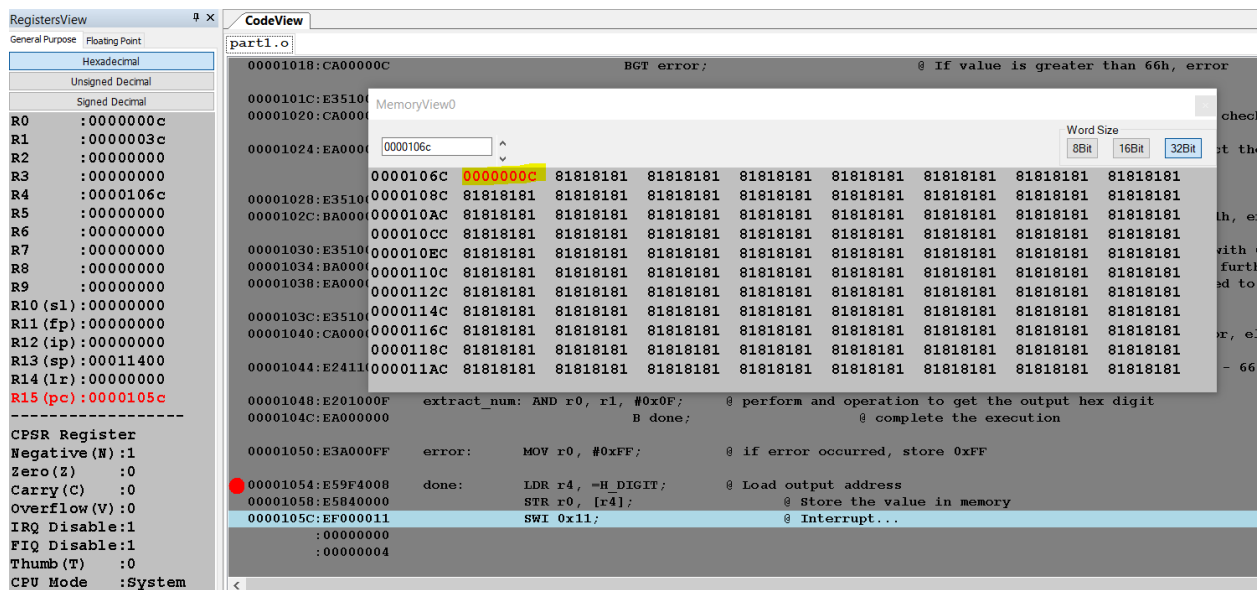
Input: ASCII is in the valid hex digit range - works for 30h-39h (numbers 0-9), 41h-46h (A-F), 61h-66h(a-f)

Note: In case a number outside this range is given as input, the output is 0xFF (ERROR).

Output: Stored in memory location 0x106c

### Test inputs and outputs:

1. Input - 0x43, Output – 0C



## 2. Input – 0x32, Output – 02

**RegistersView**

Register	Value
R0	:00000002
R1	:00000032
R2	:00000000
R3	:00000000
R4	:0000106c
R5	:00000000
R6	:00000000
R7	:00000000
R8	:00000000
R9	:00000000
R10 (s1)	:00000000
R11 (fp)	:00000000
R12 (ip)	:00000000
R13 (sp)	:00011400
R14 (lr)	:00000000
<b>R15 (pc)</b>	<b>:000105c</b>

**CPSR Register**

Negative (N)	:1
Zero (Z)	:0
Carry (C)	:0
Overflow (V)	:0
IRQ Disable	:1
FIQ Disable	:1
Thumb (T)	:0
CPU Mode	:System

**CodeView**

part1.o

00001018:CA00000C BGT error; @ If value is greater than 66h, error

0000101C:E3510000 MemoryView0

00001020:CA000000

00001024:EA000000

0000106C:00000002 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000108C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000010AC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000010CC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000010EC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000110C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000112C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000114C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000116C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000118C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000011AC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001048:E201000F extract\_num: AND r0, r1, #0x0F; @ perform and operation to get the output hex digit

0000104C:EA000000 B done; @ complete the execution

00001050:E3A000FF error: MOV r0, #0xFF; @ if error occurred, store 0xFF

00001054:E59F4008 done: LDR r4, =H\_DIGIT; @ Load output address

00001058:E5840000 STR r0, [r4]; @ Store the value in memory

0000105C:EF000011 SWI 0x11; @ Interrupt...

00000000

00000004

## 3. Input – 0x65, Output – 0E

**RegistersView**

Register	Value
R0	:0000000e
R1	:0000005e
R2	:00000000
R3	:00000000
R4	:0000106c
R5	:00000000
R6	:00000000
R7	:00000000
R8	:00000000
R9	:00000000
R10 (s1)	:00000000
R11 (fp)	:00000000
R12 (ip)	:00000000
R13 (sp)	:00011400
R14 (lr)	:00000000
<b>R15 (pc)</b>	<b>:000105c</b>

**CPSR Register**

Negative (N)	:0
Zero (Z)	:0
Carry (C)	:1
Overflow (V)	:0
IRQ Disable	:1
FIQ Disable	:1
Thumb (T)	:0
CPU Mode	:System

**CodeView**

part1.o

00001018:CA00000C BGT error; @ If value is greater than 66h, error

0000101C:E3510000 MemoryView0

00001020:CA000000

00001024:EA000000

0000106C:0000000e 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000108C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000010AC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000010CC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000010EC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000110C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000112C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000114C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000116C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000118C:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

000011AC:81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001048:E201000F extract\_num: AND r0, r1, #0x0F; @ perform and operation to get the output hex digit

0000104C:EA000000 B done; @ complete the execution

00001050:E3A000FF error: MOV r0, #0xFF; @ if error occurred, store 0xFF

00001054:E59F4008 done: LDR r4, =H\_DIGIT; @ Load output address

00001058:E5840000 STR r0, [r4]; @ Store the value in memory

0000105C:EF000011 SWI 0x11; @ Interrupt...

00000000

00000004

#### 4. Input – 0x3F, Output – FF (ERROR)

The screenshot displays a debugger interface with two main windows: **RegistersView** and **CodeView**.

**RegistersView:** Shows the state of various registers. The **R15 (pc)** register is highlighted in red, indicating the current instruction pointer, with a value of **0000105c**.

**CodeView:** Displays assembly code. The code includes a loop that checks for an error (0xFF) and stores it in memory. The code is as follows:

```
00001018:CA00000C      BGT error;          @ If value is greater than 66h, error
0000101C:E3510000      MOV r0, r1;         @ move input to r0
00001020:CA000000      BGT error;          @ If value is greater than 66h, error
00001024:EA000000      BNE done;           @ If not equal, branch to done
00001028:E3510000      MOV r0, r1;         @ move input to r0
0000102C:BA000000      BGT error;          @ If value is greater than 66h, error
00001030:E3510000      MOV r0, r1;         @ move input to r0
00001034:BA000000      BGT error;          @ If value is greater than 66h, error
00001038:EA000000      BNE done;           @ If not equal, branch to done
0000103C:E3510000      MOV r0, r1;         @ move input to r0
00001040:CA000000      BGT error;          @ If value is greater than 66h, error
00001044:E2411000      MOV r1, r0;         @ move r0 to r1
00001048:E201000F      AND r0, r1, #0x0F;  @ perform and operation to get the output hex digit
0000104C:EA000000      B done;             @ complete the execution
00001050:E3A000FF      MOV r0, #0xFF;      @ if error occurred, store 0xFF
00001054:E59F4008      LDR r4, -H DIGIT;    @ Load output address
00001058:E5840000      STR r0, [r4];        @ Store the value in memory
0000105C:EF000011      SWI 0x11;           @ Interrupt...
```

A **MemoryView0** window is open, showing the value **0000106c** at address **0000106c**. The word size is set to **32Bit**.

## Part 2 - Convert a given eight ASCII characters to an 8-bit binary number

Input: If the length of string is less than 8, the program throws error. If length is greater than 8, consider only the first 8 ASCII characters given. If any digit other than 30h or 31h is present, throw error

Output: Number – Stores the 8 bit number consisting of 0s and 1s.

Error – Contains 0x00 if no error, 0xFF if error occurred.

### Test inputs and outputs:

1. Input - 31,31,30,31,30,30,31,31,30, Outputs – Number(D3 – 11010011), Error (0)

```
00001018:E3A05008 MOV r5, #8; @ Using r5 as a counter, store value 8
0000101C:E59F4048 LDR r4, =STRING; @ Load starting address of string into register r4

00001020:E4943000 MOV r1, #0xFF; @ If error occurred, store 0xFF as Error
00001024:E3530000 EOR r0, r0, r0; @ Clear the value of number

00001028:BA000000 LDR r4, =NUMBER; @ Load address of NUMBER
0000102C:E3530000 STR r0, [r4]; @ Store the value in memory
00001030:CA000000 LDR r4, =ERROR; @ Load address of ERROR
00001034:E2033000 STR r1, [r4]; @ Store the value in memory
00001038:E1A00000 SWI 0x11; @ Interrupt...
0000103C:E1800000

00001040:E2555000 error: MOV r1, #0xFF; @ If error occurred, store 0xFF as Error
00001044:1AFF0000 EOR r0, r0, r0; @ Clear the value of number
00001048:0A000000 done: LDR r4, =NUMBER; @ Load address of NUMBER
0000104C:E3A010FF STR r0, [r4]; @ Store the value in memory
00001050:E0200000 LDR r4, =ERROR; @ Load address of ERROR
00001054:E59F4014 STR r1, [r4]; @ Store the value in memory
00001058:E5840000 SWI 0x11; @ Interrupt...
0000105C:E59F4010
00001060:E5841000
00001064:EF000011
00001068:0000002C
0000106C:00000000
00001070:00000000
00001074:00000024
00001078:00000028
```

2. Input – 31,31,30,31,30,37,31,31,30, Outputs – Number(0), Error (0xFF)

```
00001018:E3A05008 MOV r5, #8; @ Using r5 as a counter, store value 8
0000101C:E59F4048 LDR r4, =STRING; @ Load starting address of string into register r4

00001020:E4943000 MOV r1, #0xFF; @ If error occurred, store 0xFF as Error
00001024:E3530000 EOR r0, r0, r0; @ Clear the value of number

00001028:BA000000 LDR r4, =NUMBER; @ Load address of NUMBER
0000102C:E3530000 STR r0, [r4]; @ Store the value in memory
00001030:CA000000 LDR r4, =ERROR; @ Load address of ERROR
00001034:E2033000 STR r1, [r4]; @ Store the value in memory
00001038:E1A00000 SWI 0x11; @ Interrupt...
0000103C:E1800000

00001040:E2555000 error: MOV r1, #0xFF; @ If error occurred, store 0xFF as Error
00001044:1AFF0000 EOR r0, r0, r0; @ Clear the value of number
00001048:0A000000 done: LDR r4, =NUMBER; @ Load address of NUMBER
0000104C:E3A010FF STR r0, [r4]; @ Store the value in memory
00001050:E0200000 LDR r4, =ERROR; @ Load address of ERROR
00001054:E59F4014 STR r1, [r4]; @ Store the value in memory
00001058:E5840000 SWI 0x11; @ Interrupt...
0000105C:E59F4010
00001060:E5841000
00001064:EF000011
00001068:0000002C
0000106C:00000000
00001070:00000000
00001074:00000024
00001078:00000028
```

### 3. Input – 31,31,30,31,30,24,31,31,30, Outputs – Number(0), Error (0xFF)

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 00000000

R1 : 000000ff

R2 : 00000009

R3 : 00000024

R4 : 000010a0

R5 : 00000003

R6 : 00000000

R7 : 00000000

R8 : 00000000

R9 : 00000000

R10 (s1): 00000000

R11 (fp): 00000000

R12 (ip): 00000000

R13 (sp): 00011400

R14 (lr): 00000000

**R15 (pc): 00001064**

-----

CPSR Register

Negative (N): 1

Zero (Z): 0

Carry (C): 0

Overflow (V): 0

IRQ Disable: 1

FIQ Disable: 1

Thumb (T): 0

CPU Mode : System

CodeView

part2.o

00001018: E3A05008 MOV r5, #0; @ Using r5 as a counter, store value 0

0000101C: E59F4048 LDR r4, -STRING; @ Load starting address of string into register r4

MemoryView0

00001020: E4943000 0000109c 00000000 000000ff 00000009 81818181 81818181 81818181 81818181 81818181

00001024: E3530000 000010BC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001028: BA000000 000010DC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000102C: E3530000 000010FC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001030: CA000000 0000111C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001034: E2033000 0000113C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001038: E1A00000 0000115C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000103C: E1800000 0000117C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001040: E2555000 0000119C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001044: 1AFFFD00 000011BC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001048: 0A000000 000011DC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000104C: E3A010FF error: MOV r1, #0xFF; @ If error occurred, store 0xFF as Error

00001050: E0200000 EOR r0, r0, r0; @ Clear the value of number

00001054: E59F4014 done: LDR r4, -NUMBER; @ Load address of NUMBER

00001058: E5840000 STR r0, [r4]; @ Store the value in memory

0000105C: E59F4010 LDR r4, -ERROR; @ Load address of ERROR

00001060: E5841000 STR r1, [r4]; @ Store the value in memory

00001064: EF000011 SWI 0x11; @ Interrupt...

### 4. Input – 31,30,31,31, Outputs – Number(0), Error (0xFF)

RegistersView

General Purpose Floating Point

Hexadecimal

Unsigned Decimal

Signed Decimal

R0 : 00000000

R1 : 000000ff

R2 : 00000005

R3 : 00000000

R4 : 00001090

R5 : 00000000

R6 : 00000000

R7 : 00000000

R8 : 00000000

R9 : 00000000

R10 (s1): 00000000

R11 (fp): 00000000

R12 (ip): 00000000

R13 (sp): 00011400

R14 (lr): 00000000

**R15 (pc): 00001064**

-----

CPSR Register

Negative (N): 1

Zero (Z): 0

Carry (C): 0

Overflow (V): 0

IRQ Disable: 1

FIQ Disable: 1

Thumb (T): 0

CPU Mode : System

CodeView

part2.o

00001018: E3A05008 MOV r5, #0; @ Using r5 as a counter, store value 0

0000101C: E59F4048 LDR r4, -STRING; @ Load starting address of string into register r4

MemoryView0

00001020: E4943000 0000108c 00000000 000000ff 00000005 81818181 81818181 81818181 81818181 81818181

00001024: E3530000 000010AC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001028: BA000000 000010CC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000102C: E3530000 000010EC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001030: CA000000 0000110C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001034: E2033000 0000112C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001038: E1A00000 0000114C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000103C: E1800000 0000116C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001040: E2555000 0000118C 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001044: 1AFFFD00 000011AC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

00001048: 0A000000 000011CC 81818181 81818181 81818181 81818181 81818181 81818181 81818181 81818181

0000104C: E3A010FF error: MOV r1, #0xFF; @ If error occurred, store 0xFF as Error

00001050: E0200000 EOR r0, r0, r0; @ Clear the value of number

00001054: E59F4014 done: LDR r4, -NUMBER; @ Load address of NUMBER

00001058: E5840000 STR r0, [r4]; @ Store the value in memory

0000105C: E59F4010 LDR r4, -ERROR; @ Load address of ERROR

00001060: E5841000 STR r1, [r4]; @ Store the value in memory

00001064: EF000011 SWI 0x11; @ Interrupt...

### Part 3 - Convert given 8 digit packed BCD number to 32 bit number

Input represents the given number in hex format which are 8 digits of packed BCD number (4 bits for each digit)

Shift right by 4 each time and obtain the lower 4 bits each time

Find output by multiplying by 1,10, 100, 1000.. starting from the least significant nibble and accumulating the result

#### Test inputs and outputs:

Input – 0x92529679

Output – 0x0583e40f

```
RegistersView
General Purpose Floating Point
Hexadecimal
Unsigned Decimal
Signed Decimal
R0 :0583e40f
R1 :055d4a80
R2 :00000000
R3 :05f5e100
R4 :0000104c
R5 :00000000
R6 :00000000
R7 :00000000
R8 :00000000
R9 :00000000
R10 (s1):00000000
R11 (fp):00000000
R12 (ip):00000000
R13 (sp):00011400
R14 (lr):00000000
R15 (pc):0000103c
-----
CPSR Register
Negative (N):0
Zero (Z):1
Carry (C):1
Overflow (V):0
IRQ Disable:1
FIQ Disable:1
Thumb (T):0
CPU Mode :System
-----
0x6000004f

MemoryView0
part
Word Size
8Bit 16Bit 32Bit
0000104c 0583E40F 81818181 81818181 81818181 81818181 81818181 81818181
0000106C 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000108C 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010AC 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010CC 81818181 81818181 81818181 81818181 81818181 81818181 81818181
000010EC 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000110C 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000112C 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000114C 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000116C 81818181 81818181 81818181 81818181 81818181 81818181 81818181
0000118C 81818181 81818181 81818181 81818181 81818181 81818181 81818181
00001010:E3A03001 MOV r3, #1; @ Initialize r3 to 1 initially, this holds t
00001014:E3A0400A MOV r4, #10; @ Register r4 always contains the value 10
00001018:E202100F loop: AND r1, r2, #0x0F; @ Obtain lower 4 bits by this operation
0000101C:E0010391 MUL r1, r1, r3; @ Multiply these 4 bits by the appropriate power of
00001020:E0800001 ADD r0, r0, r1; @ Accumulate the result in r0
00001024:E0030493 MUL r3, r3, r4; @ Increase the multiplicand by a factor of 10 each t
00001028:E1A02222 MOV r2, r2, LSR #4; @ Shift the number right by 4 bits to get th
0000102C:E3520000 CMP r2, #0; @ Check if all the bits have been shifted
00001030:1AFFFFF8 BNE loop; @ Continue until all bits are read
00001034:E59F4008 done: LDR r4, =NUMBER; @ Load address of output NUMBER
00001038:E5840000 STR r0, [r4]; @ Store the value in memory
0000103C:EF000011 SWI 0x11; @ Interrupt...
:00000000
:00000004
```