| TUTORIAL #1 | Data and Instruction Representation | (CS1006) |
|---|---|---|

## 1.1 Number representation

(To be done over 2 week)

(1) Give the *decimal* equivalent value of the 8-bit binary numbers using both the ***unsigned magnitude*** and ***2's complement*** number representations.

The binary numbers are:

(a) **$0111\ 1111_2$** (b) **$1111\ 1111_2$** (c) **$0000\ 0000_2$**
(d) **$1000\ 0000_2$** (e) **$1111\ 1110_2$**

NOTE: If you wish, you can use this website or any other equivalent to check your answer:
https://www.rapidtables.com/convert/number/binary-to-decimal.html

(2) For each of the number representation in part (1), determine its range by stating what is the **largest** and **smallest** decimal numbers you can represent using only eight binary digits.

(3) Give the numeric range of variables declared with the following ANSI C data types:
(a) `unsigned char` (b) `short int`
(c) `unsigned short int` (d) `long int`

(4) What would be the **most efficient** ANSI C data type to assign to a variable in your C program that represents the following:
(a) current temperature (°C) as a whole number at any selected city in the world.
(b) total undergraduate population in NTU at any given moment.
(c) current total US national debt in US$ (check: http://www.usdebtclock.org/ ).
(d) whether a person is male or female.

## 1.2 Hexadecimal number representation

Below are 8-bit hexadecimal numbers:

(a) **0xFF** = 1111 1111

(b) **0x0F** = 0000 1111

(c) **0x4D** = 0100 1101

(d) **0xC0** = 1100 0000

(e) **0x30** = 0011 0000

**Note:** the **'0x'** prefix is used to signify that the number is in hexadecimal notation.

| LS \ MS | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **0** | NUL | DLE | SP | 0 | @ | P | ' | p |
| **1** | SOH | DC1 | ! | 1 | A | Q | a | q |
| **2** | STX | DC2 | " | 2 | B | R | b | r |
| **3** | ETX | DC3 | # | 3 | C | S | c | s |
| **4** | EOT | DC4 | $ | 4 | D | T | d | t |
| **5** | ENQ | NAK | % | 5 | E | U | e | u |
| **6** | ACK | SYN | & | 6 | F | V | f | v |
| **7** | BEL | ETB | ' | 7 | G | W | g | w |
| **8** | BS | CAN | ( | 8 | H | X | h | x |
| **9** | HT | EM | ) | 9 | I | Y | i | y |
| **A** | LF | SUB | * | : | J | Z | j | z |
| **B** | VT | ESC | + | ; | K | [ | k | { |
| **C** | FF | FS | , | < | L | \ | l | \| |
| **D** | CR | GS | - | = | M | ] | m | } |
| **E** | SO | RS | . | > | N | ^ | n | ~ |
| **F** | SI | US | / | ? | O | _ | o | DEL |

**ASCII Character Set (7-Bit Code)**

(1) Which of these numbers are negative in 2's complement representation? a, d

(2) Which of these numbers are valid ASCII alpha-numeric characters? b, c, e

(3) Give the decimal equivalent values of the 2's complement numbers in (a) and (b)? 1, 15

(4) Would the 32-bit hexadecimal values of **0xFFFFFFFF** and **0x0000000F** have the same decimal value (*in 2's complement*) as the corresponding 8-bit values **0xFF** and **0x0F** given in (a) and (b) respectively?

(5) With reference from part (4), device a simple technique to convert any 8-bit sized number to its 32-bit sized equivalent number without changing its 2's complement value (e.g. convert the decimal value **-3** given by the 8-bit hexadecimal value **0xFD** to its 32-bit equivalent).

(1) Give the *decimal* equivalent value of the 8-bit binary numbers using both the ***unsigned magnitude*** and ***2's complement*** number representations.

The binary numbers are: 00000001

(a) **0111 1111**$_2$ (b) **1111 1111**$_2$ (c) **0000 0000**$_2$

(d) **1000 0000**$_2$ (e) **1111 1110**$_2$ 0 0010

a) 127/ 127 b) 255/-1 c) 0/0

d) 128/-128 e) 254/-2

(2) For each of the number representation in part (1), determine its range by stating what is the **largest** and **smallest** decimal numbers you can represent using only eight binary digits.

| | small | big |
|---|---|---|
| unsign | 0 | 255 |
| sign | -128 | 127 |

(3) Give the numeric range of variables declared with the following ANSI C data types:

(a) `unsigned char` (b) `short int`

(c) `unsigned short int` (d) `long int`

3a) 1 byte, 0-256 b) -32768-32767

c) 0-65536 d) -2147483648 - 2147483647

(4) What would be the **most efficient** ANSI C data type to assign to a variable in your C program that represents the following:

(a) current temperature (°C) as a whole number at any selected city in the world.

(b) total undergraduate population in NTU at any given moment.

(c) current total US national debt in US$ (check: http://www.usdebtclock.org/ ).

(d) whether a person is male or female.

4a) char

b) unsigned int

c)

d) - bool

## 1.3 Data representation in memory

Various variables and constants of different ANSI C data types have been declared and stored in memory. Figure 1.3 shows the byte-sized contents in memory where they can be found. Based on these memory contents, answer the following questions:

| Address | Contents | |
|---|---|---|
| 0x0000 | 0x08 | |
| 0x0001 | 0x35 | E |
| 0x0002 | 0xFF | / |
| 0x0003 | 0x01 | / |
| 0x0004 | 0xA8 | / |
| 0x0005 | 0x2A | * |
| 0x0006 | 0x4C | L |
| 0x0007 | 0x6F | o |
| 0x0008 | 0x67 | g |
| 0x0009 | 0x69 | i |
| 0x000A | 0x6E | n |
| 0x000B | 0x3A | : |
| 0x000C | 0x00 | / |
| 0x000D | 0x61 | a |
| 0x000E | 0x62 | b |
| 0x000F | 0x63 | c |

**Figure 1.3 – Contents in memory**

(1) Find the start address of a 2-byte integer with the decimal value of 511. Is this integer in big or little endian format?

0x1FF, 0x0002, little endian

(2) Find a possible C string among the memory contents. "* Login: "

(3) A structure and variable declaration is given below:

```
struct rec {
   unsigned char i;   1 byte
   long int j;   4 byte
   char a[3];   4 byte
};
struct rec r;
```

Assume the starting address of variable **r** is 0x0008 and the **big endian** format has been adopted, give (in hexadecimal) the values of following:

(a) **r.i** (b) **r.j** (c) **r.a[0]** (d) **r.a[2]**

0x0008 0x0012 0x0016 0x0018

Note: Assume no data alignment is required for multi-byte variables

## 1.4 Data and Address Busses

Figure 1.4 shows the pin out of the MC68000 microprocessor. The data pins are labeled **Dn** and the address pins are labeled **An**.

(1) Based on the address pin labeling, what is the maximum memory capacity addressable by this processor (in Mbytes)? Assume the missing address pin **A0** can be derived from the **UDS*** and **LDS*** pins.

24 pin, $2^{24}/2^{20}$ = 16 MB

(2) Based on the data pin labeling, what is the maximum number of bytes can this processor transfer within one memory cycle?

16 Pin, 2 bytes

(3) With reference to the data structure in Question 1.3 part (3), re-design the structure `rec` to make it more efficient for use within a computing system supported by a MC68000 processor? Give a reason for your re-design.

```
struct rec {
  long int j;
  unsign char i;
  char a[3]
}
struct rec r
```

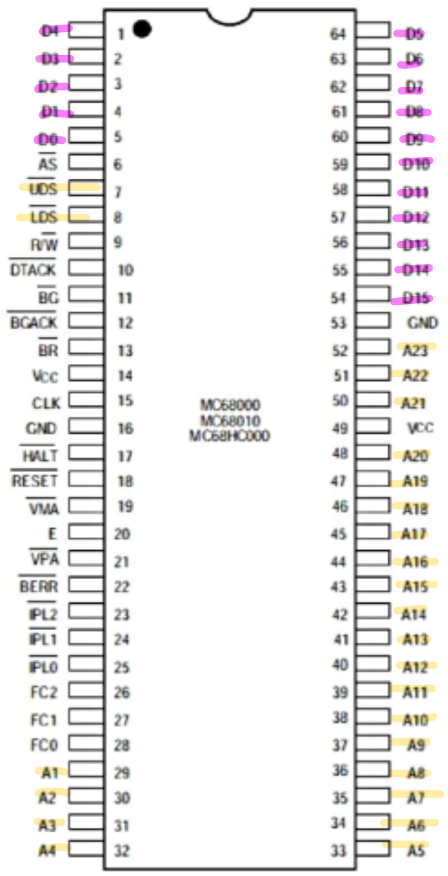


**Figure 1.4 – MC68000 processor pin out**

## 1.5 ARM Programmer's Model and Instruction Execution

Figures 1.5 show the display of the VisUAL ARM simulator and the ARM assembly program example "*Tutorial_1_5*" found in the NTULearn Tutorial 1 folder.

**Note:** You can download the VisUAL ARM emulator from: https://salmanarif.bitbucket.io/visual/

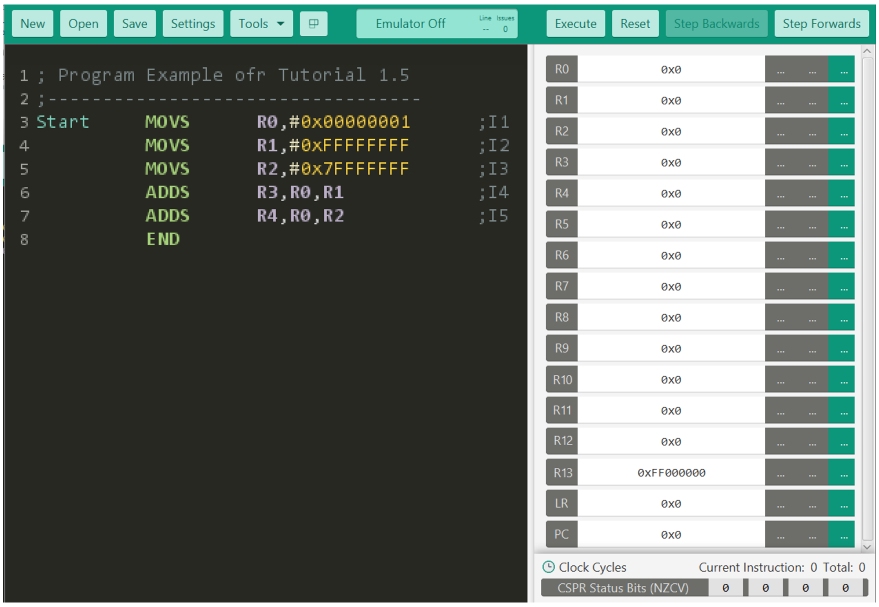


**Figure 1.5 – View on the VisUAL ARM simulator after loading the program "*Tutorial_1_5*".**

(1) What size (in number of bits) are the registers in the ARM processor? 32 bit

(2) With reference to question 1.1 part (2), state what are the **largest** and **smallest hexadecimal values** you can find in register **R0** if you are using ***unsigned magnitude*** and ***2's complement*** number representations interpretation.

| | | |
|---|---|---|
| L | 0x FFFFFFFF | 0x 7FFF FFFF |
| S | 0x 0000 0000 | 0x 8000 0000 |

(3) What are the available registers shown in VisUAL user interface in Figure 1.5? Is this the complete set of usable registers in the ARM User Mode Programmer's Model?

(4) Briefly describe what each of the ARM instructions will do when they are executed?

(5) Will executing `MOVS R1,#0xFFFFFFFF` change the state of any of the N, Z, V, C flags in the Current Program Status Register (CPSR)? If so, why?

(6) Will executing `MOVS R2,#0x7FFFFFFF` immediately after the instruction in part (5) change the state of any of the N, Z, V, C flags? If so, why?

(7) The instruction `ADDS R3,R0,R1` adds the 32-bit content in registers **R0** and **R1** and put the result into the destination register **R3**. Answer the following questions related to this instruction:

a) Give the 32-bit result in destination register **R3** after the execution of this instruction? 0x0
b) Is the answer correct for the addition of the numbers `0x00000001` and `0xFFFFFFFF`? no
c) What are the values of the N, Z, V, C flags immediately after the execution of this instruction? 0101
d) Explain the reason why each of the flags are set.

(8) Now do the same as in part (7) but now for the instruction `ADDS R4,R0,R2`. What can you say is the difference between the interpretation of C and V flags when they are set?

(9) Give an example of two 32-bit numbers when added will set both the C and V flags simultaneously? **Note:** You can test to see if your answer is correct by editing the *Tutorial_1_5* program.

0x8000 000
0x8000 001