UE719

Embedded Systems Tutorials : ARM Part

Lamri NEHAOUA

lamri.nehaoua@univ-evry.fr

November 15, 2022

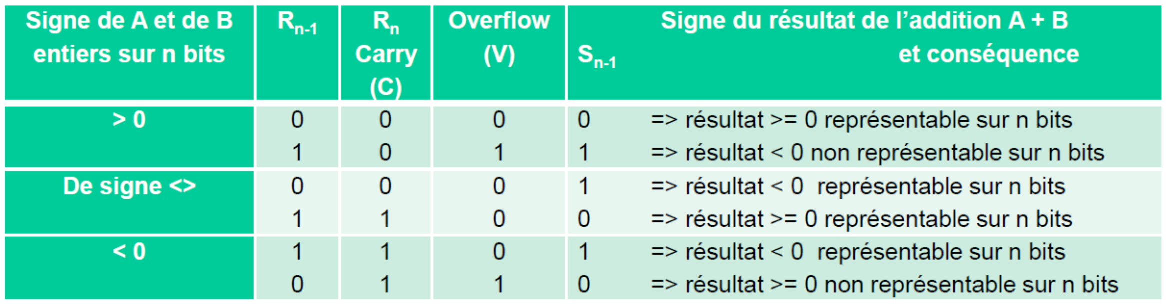
CONTENTS

1 PROCESSOR ARITHMETIC

1. For each operation, indicate the status of the obtained result. Express this state by using the correspondig flag: Z, C, N, V.
   (a) Natural numbers: 0x34FF + 0x1221 and 0x34FF + 0xCB01
   (b) Perform the following addition 0x67AB + 0x187F on an 8 bit processor. We consider an addition instruction without carry flag for the least significant bytes and an addition instruction with carry flag for the most significant bytes.
   (c) Integer numbers: 0x85 + 0xA3, 0xC5 + 0x63, 0x45 + 0x63, 0x5 + 0x23.
2. Represent the two 8 bit integer 0x23 and 0x85 on 32 bit format.
3. Consider the two 8 bit integer 0x20 and 0x80. Apply two logic shift and two arithmetic shift. Conclude.

A result of an addition of two n-bit integers is correct and hence representable on n bits if: $R_n = R_{n-1}$ and $C = R_{n-1}$. For flag V, overflow:

- $V = 0$ if $C = R_{n-1}$
- $V = 1$ if $C \neq R_{n-1}$



Signe de A et de B entiers sur n bits	R_{n-1}	R_n Carry (C)	Overflow (V)	S_{n-1}	Signe du résultat de l'addition A + B et conséquence
> 0	0	0	0	0	=> résultat >= 0 représentable sur n bits
	1	0	1	1	=> résultat < 0 non représentable sur n bits
De signe <>	0	0	0	1	=> résultat < 0 représentable sur n bits
	1	1	0	0	=> résultat >= 0 représentable sur n bits
< 0	1	1	0	1	=> résultat < 0 représentable sur n bits
	0	1	1	0	=> résultat >= 0 non représentable sur n bits

2 BINARY MASK

Consider a register containing the value 0xFA59. Suggest logical operations allowing to:

- To set the 8 most significant bits to '0',
- To set the 8 least significant bits to '1',
- To highlight only the most significant bit,
- To reset only the least significant bit.

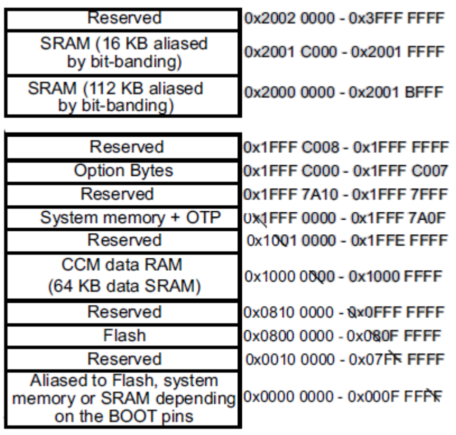
3 MEMORY MAPPING STM32F407

The following figure is an overview of the memory map from the documentation of the STM32F407 microcontroller. By analyzing it, determine the following:

1. What is the maximum size of the address bus?

2. Identify the area corresponding to the program memory, give its first and last element adresses and calculate its size.

3. Identify the areas corresponding to the data memory, give their first and last element adresses and calculate the total size.



Reserved	0x2002 0000 - 0x3FFF FFFF
SRAM (16 KB aliased by bit-banding)	0x2001 C000 - 0x2001 FFFF
SRAM (112 KB aliased by bit-banding)	0x2000 0000 - 0x2001 BFFF
Reserved	0x1FFF C008 - 0x1FFF FFFF
Option Bytes	0x1FFF C000 - 0x1FFF C007
Reserved	0x1FFF 7A10 - 0x1FFF 7FFF
System memory + OTP	0x1FFF 0000 - 0x1FFF 7A0F
Reserved	0x1001 0000 - 0x1FFE FFFF
CCM data RAM (64 KB data SRAM)	0x1000 0000 - 0x1000 FFFF
Reserved	0x0810 0000 - 0x0FFF FFFF
Flash	0x0800 0000 - 0x080F FFFF
Reserved	0x0010 0000 - 0x07FF FFFF
Aliased to Flash, system memory or SRAM depending on the BOOT pins	0x0000 0000 - 0x000F FFFF

4 MOV INSTRUCTION AND LDR PSEUDO-INSTRUCTION

1. Indicate the contents of the involved register at the end of the execution of each instruction.

2. Review the generated code by de-assembling the pseudo-instruction LDR R3, =0x40000000.

3. The instruction LDR R5, 0x40000001 was replaced by the assembler by an indirect addressing instruction using the PC register and an offset (PC + 4). Determine the address of the location of the constant loaded in R5.

```
; Main Program
; ************************************************************
        AREA my_program, CODE, READONLY
value1    EQU 0x23AF
value2    EQU 0xFF23DE62
          ENTRY
          EXPORT __main
__main
         ; IMMEDIAT addressing
         ; ************************
         ; ************************
         MOV    R0, #0xEF          ;load immediat 8 bit constant
         MOV    R1, #0x7F32        ;load immediat 16 bit constant
         MOV    R2, #valeur1       ;load immediat 16 bit constant

         ; load immediat constant over 16 bit
         ; *******************************************************
         LDR    R3, =0x40000000 ; 16 bits encoded
         MOV.W  R4, #0x40000000 ; the same thing
         LDR    R5, =0x40000001 ; can't be encoded in 16 bits
         NOP
boucle   B      boucle
         END
```

5 BIT C, BIT V WHAT TO CHOOSE?

1. Manually perform the operation indicated in the code bellow and set the bits of the status register CPSR.

2. Conclude on the quality of the result if we consider that:

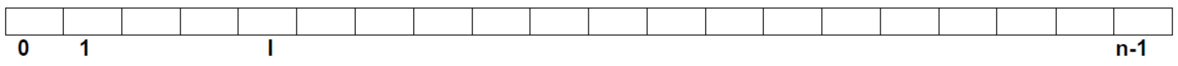
   Value1 and Value2 are natural

   Value1 and Value2 are integer

```
; Main Program
; ***********************************************************
        AREA my_program, CODE, READONLY
value1    EQU 0xFF000001
value2    EQU 0xFF000002
          ENTRY
          EXPORT __main
__main
         ; IMMEDIAT addressing
         ; *************************
         ; *************************
         MOV    R2, #valeur1
         MOV    R2, #valeur2
         ADDS   R2, R0, R1
         NOP
boucle   B      boucle
         END
```

6 1D ARRAY

A one-dimensional array is a contiguous list of elements of the same type, each element is located by its index I and in memory, it is located by its address. Example of an array of n elements:

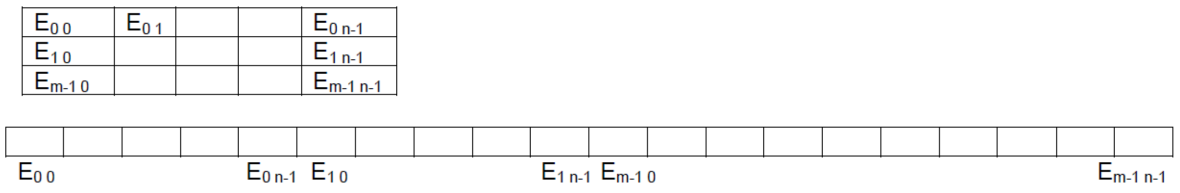


A 1D array is implemented in memory in the form of n consecutive locations. This memory area is defined by a start address (the element with index $I = 0$). The overall size of the occupied memory area is dependent on the size of an element and the number of elements.

1. Propose a general formula for calculating the address of an element of index I of the array. Application: base address = 0x2000 0000, element size 32 bits, nimber of elements 24.
2. Calculate the memory size occupied by the array.
3. Calculate the address of the element with index 10.
4. Calculate the address of the first memory element located just after the previous array.

7 2D ARRAY

A two-dimensional array is also a contiguous list of elements of the same type row by row or column by column, the example below illustrates an $m \times n$ array and its row/row layout:



1. Propose a general formula for calculating the address of an element of index $[I, J]$ of the array. Application: base address = 0x2000 0000, element size 16 bits, nimber of elements 4 rows and 10 columns.

2. Calculate the memory size occupied by the array.

3. Calculate the address of the element $[2, 7]$.

4. Calculate the address of the first memory element located just after the previous array.

8 INDIRECT ADDRESSING

Indicate the contents of the involved register at the end of the execution of each instruction.

```
; Main Program
; *************************************************************
             AREA my_constant, CODE, READONLY
desoctets    DCB 0x10, 0xFF, 34, 125, 'a', 0x2F, 'w', 255
des32bits    DCD 0x55AA55AA, 0X10356278, 0x23242526
             AREA my_program, CODE, READONLY
             ENTRY
             EXPORT __main
__main
         ; INDIRECT addressing
         ; *************************
         ; *************************
         LDR   R0, =desoctets
         LDR   R1, [R0]
         LDRB  R2, [R0]
         LDRH  R3, [R0]
         LDRSH R4, [R0]
         NOP

            ; PRE and POST INDEXED addressing
        ; *****************************************
        LDR   R0, =des32bits
        MOV   R1, R0
        LDR   R2, [R0, #4]!
        LDR   R3, [R0, #4]!
        LDR   R4, [R1], #4
        LDR   R5, [R1], #4

         NOP
boucle   B    boucle
         END
```