Microprocessor Principles and Applications –Midterm (Hands-on Test)

Nov. 5, 2020

1. Implement a sorting algorithm to sort the given five 8-bit numbers.
   (a) (25%) Write a macro called INITIALIZE(arg1, arg2, arg3, arg4, arg5) to locate the given five 8-bit numbers (e.g., 0xA5, 0xF4, 0x64, 0x6F, 0x87) at 0x100 to 0x104 in memory. You are required to use at least one kind of register regarding addressing mode.
   (b) (25%) Write a subroutine called SORTING to implement the sorting algorithm and store the sorting results at 0x100 to 0x104 in memory.

Note: You have to allocate memory for the five numbers by yourself and we may change those numbers when you demonstrate to us.

2. (50%) Write a program to compute the product of two 16-bit signed integers. You should store high byte of the first number in [0x00], the low byte of the first number in [0x01], high byte of the second number in [0x02], low byte of the second number in [0x03] ,and store the result in [0x04 to 0x07]..

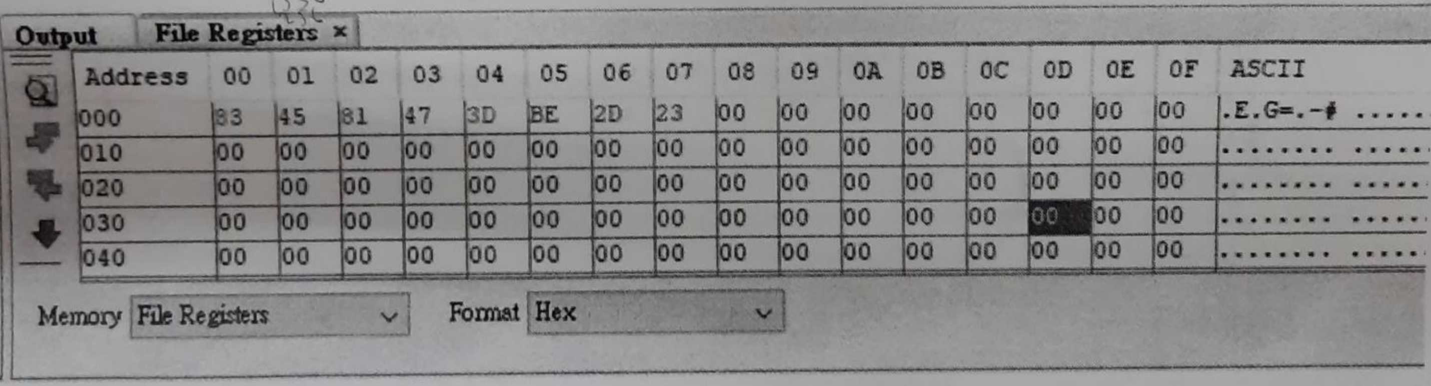
EXAMPLE:

*8345_{16} * 8147_{16} (-3193_{10} * -32441_{10})* = $3DBE2D23_{16}$($103587357 1_{10}$)

$1FF4_{16}$ * $82C4_{16}$(8180_{10} * -32060_{10})= $F05E5ED0_{16}$ (-262250800_{10})

256
16
1536
1536

96 40 9 6



Output | File Registers ×

Address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	ASCII
000	83	45	81	47	3D	BE	2D	23	00	00	00	00	00	00	00	00	.E.G=.-# .....
010	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

Memory File Registers

Format Hex

You can refer to the following steps:

STEP1. multiply 2 operands disregarding their signs (hint: mulwf,addwf,addwfc)

- compute (low byte of the first number) * (low byte of the second number)
  - store the low byte of the result in [0x07]
  - store the high byte of the result in [0x06]
- compute (high byte of the first number) * (high byte of the second number)
  - store the high byte of the result in [0x05]
  - store the low byte of the result in [0x04]
- compute (low byte of the first number) * (high byte of the second number)

  - add the result to [0x06]
  - add the result and carry to [0x05]
  - add carry to [0x04]
- compute (high byte of the first number) * (low byte of the second number)
  - add the result to [0x06]
  - add the result and carry to [0x05]
  - add carry to [0x04]

STEP2. check the sign of the first number (hint: subwf,subwfb)

If negative, then subtract the first operand from the **upper 16 bits** of the product.

STEP3. check the sign of the second number (hint: subwf,subwfb)

If negative, then subtract the second operand from the **upper 16 bits** of the product.

10000 0000 1011 0000

0

82C4 1111 0000 0101 1110

0001 0000 0101 0010

1052

0

~~1011 1110 1010 0011~~

~~BE A3~~ 1000 0010 1100 0100

82C4 1111 0000 0101 1110

F05E

ADDWFC ADD W and Carry bit to f

Syntax:	[*label*] ADDWFC f [,d [,a]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(W) + (f) + (C) → dest
Status Affected:	N,OV, C, DC, Z
Encoding:	0010 00da ffff ffff
Description:	Add W, the Carry Flag and data memory location 'f'. If 'd' is 0, the result is placed in W. If 'd' is 1, the result is placed in data memory location 'f'. If 'a' is 0, the Access Bank will be selected. If 'a' is 1, the BSR will not be overridden.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `ADDWFC REG, 0, 1`

Before Instruction

Carry bit	=	1
REG	=	0x02
W	=	0x4D

After Instruction

Carry bit	=	0
REG	=	0x02
W	=	0x50

SUBWFB Subtract W from f with Borrow

Syntax:	[*label*] SUBWFB f [,d [,a]
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – (W) – ($\overline{C}$) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	0101 10da ffff ffff
Description:	Subtract W and the carry flag (borrow) from register 'f' (2's complement method). If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example 1: `SUBWFB REG, 1, 0`

Before Instruction

REG	=	0x19	(0001 1001)
W	=	0x0D	(0000 1101)
C	=	1	

After Instruction

REG	=	0x0C	(0000 1011)
W	=	0x0D	(0000 1101)
C	=	1	
Z	=	0	
N	=	0	; result is positive

Example 2: `SUBWFB REG, 0, 0`

Before Instruction

REG	=	0x1B	(0001 1011)
W	=	0x1A	(0001 1010)
C	=	0	

After Instruction

REG	=	0x1B	(0001 1011)
W	=	0x00	
C	=	1	
Z	=	1	; result is zero
N	=	0	

Example 3: `SUBWFB REG, 1, 0`

Before Instruction

REG	=	0x03	(0000 0011)
W	=	0x0E	(0000 1101)
C	=	1	

After Instruction

REG	=	0xF5	(1111 0100) ; [2's comp]
W	=	0x0E	(0000 1101)
C	=	0	
Z	=	0	
N	=	1	; result is negative