Midterm Hands-on Exam   Microprocessor Principles and Applications, Fall 2018   11/19/2018

1. **Only MPASM assembly language is allowed**
2. **Get 0 points, if any ways of cheating**
3. **NO communication devices or applications allowed**
4. **Examination time: 1800-2100**

Implement a 16-bit Restricted divider and display the results

Description:

1. Build a 16-bit Divider. (Please refer to Figure 1, Figure 2, and Table 1)
   - Dividend size is 16 bits.
   - Divisor size is 8 bits. (Range: **1 <= Divisor value <= 255, NO ZERO!**)
   - Quotient size is 8 bits. (Range: **0 <= Quotient value <= 255**) (The result I want)
   - Remainder size is 8 bits. (The result I want)
2. Split your results to 2 values from the bit7-bit4 to bit3-bit0 and put values on 2 file registers. (Please refer to Figure 3)

3. Pushing button 1 can display the result of Quotient. Pushing button 2 can display the result of Remainder. LED status is dark for 0.5 seconds and LED status is bright for 1 second. (Please refer to Figure 4)

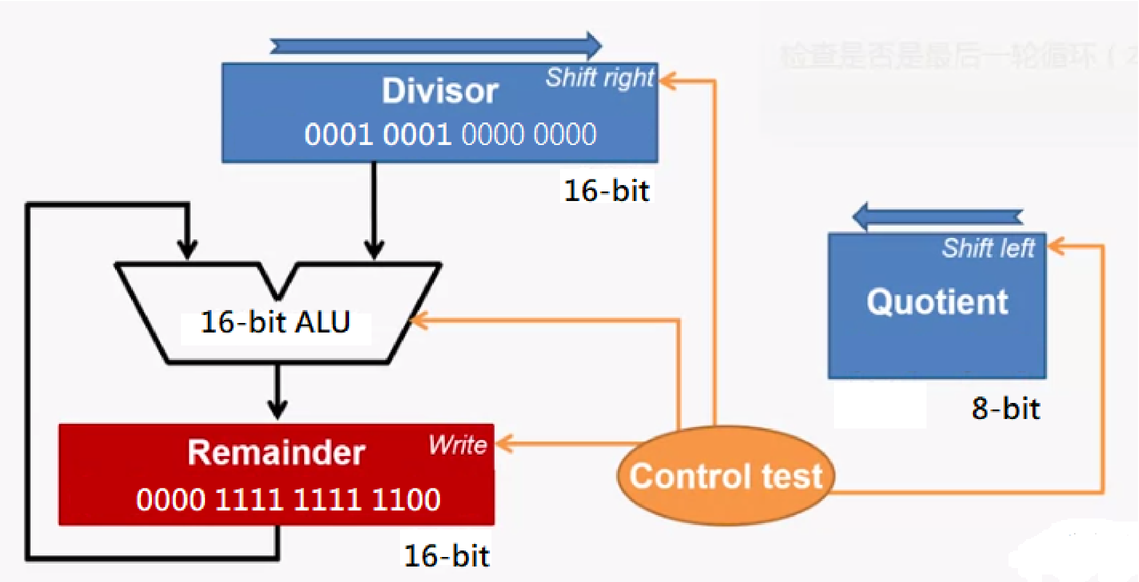


Figure 1

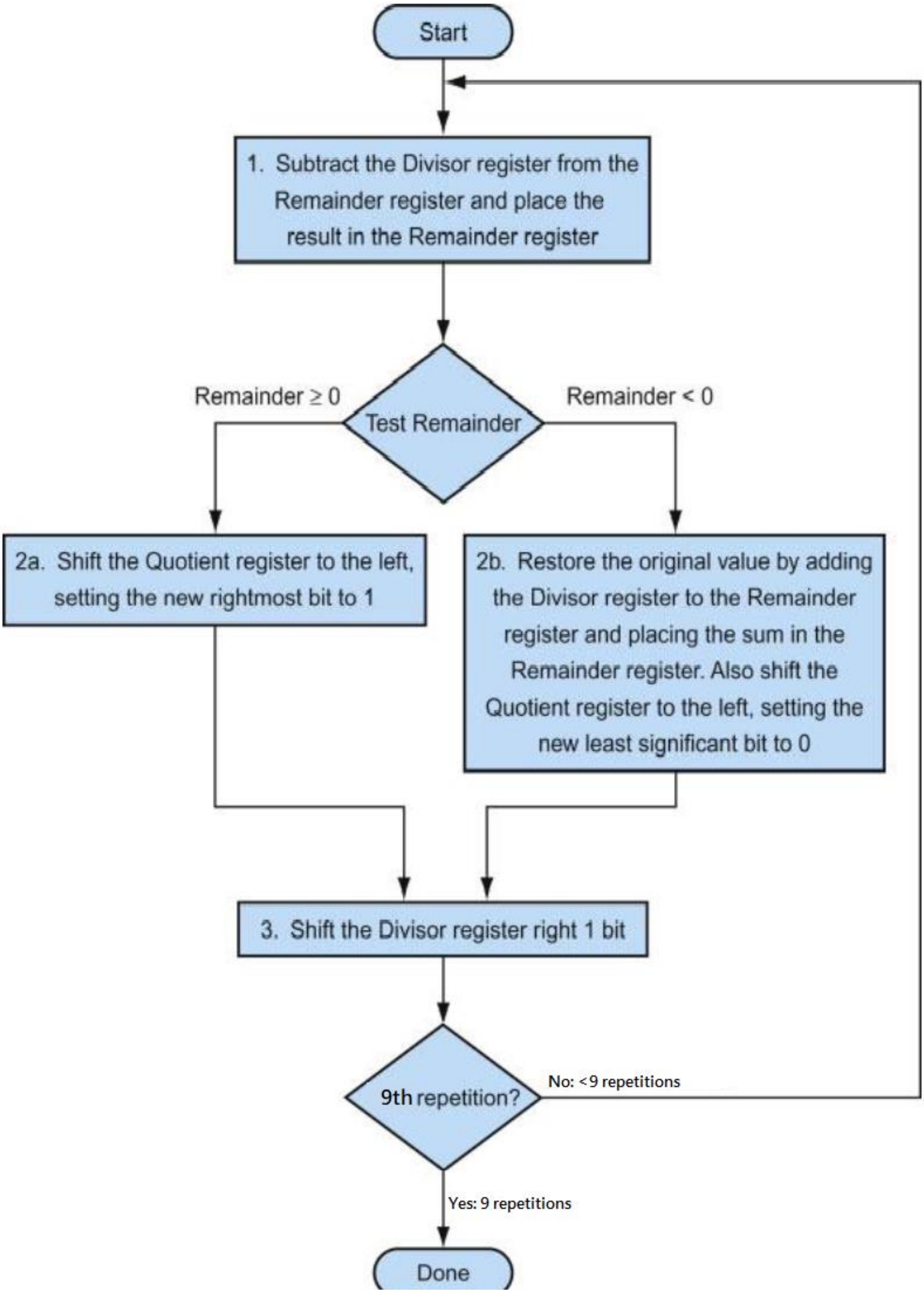
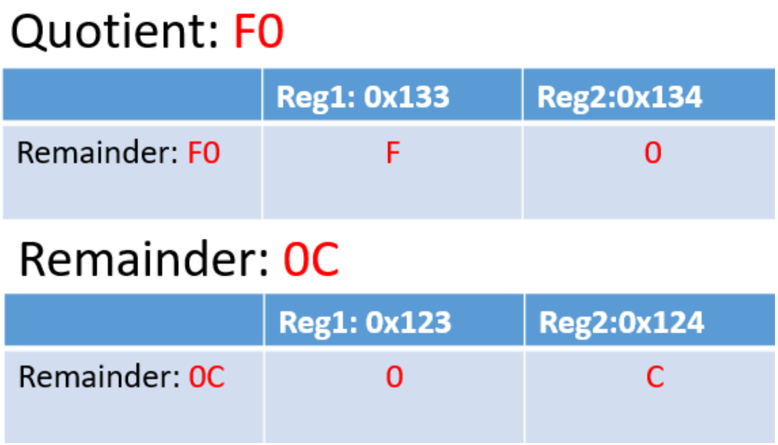

Start
1. Subtract the Divisor register from the Remainder register and place the result in the Remainder register
Remainder ≥ 0
Test Remainder
Remainder < 0
2a. Shift the Quotient register to the left, setting the new rightmost bit to 1
2b. Restore the original value by adding the Divisor register to the Remainder register and placing the sum in the Remainder register. Also shift the Quotient register to the left, setting the new least significant bit to 0
3. Shift the Divisor register right 1 bit
9th repetition?
No: <9 repetitions
Yes: 9 repetitions
Done

Figure 2

		Quotient	Divisor	Remainder
		0	0001000100000000	0000111111111100
1	Remainder-Divisor<0	0	0001000100000000	0000111111111100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	0	0001000100000000	0000111111111100
	Divisor Rotates right 1 bit.	0	0000100010000000	0000111111111100
2	Remainder-Divisor>=0, Remainder = Remainder - Divsior	0	0000100010000000	0000011101111100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	1	0000100010000000	0000011101111100
	Divisor Rotates right 1 bit.	1	0000010001000000	0000011101111100
3	Remainder-Divisor>=0, Remainder = Remainder - Divsior	1	0000010001000000	0000001100111100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	11	0000010001000000	0000001100111100
	Divisor Rotates right 1 bit.	11	0000001000100000	0000001100111100
4	Remainder-Divisor>=0, Remainder = Remainder - Divsior	11	0000001000100000	0000000100011100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	111	0000001000100000	0000000100011100
	Divisor Rotates right 1 bit.	111	0000000100010000	0000000100011100
5	Remainder-Divisor>=0, Remainder = Remainder - Divsior	111	0000000100010000	0000000000001100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	1111	0000000100010000	0000000000001100
	Divisor Rotates right 1 bit.	1111	0000000010001000	0000000000001100
6	Remainder-Divisor<0	1111	0000000010001000	0000000000001100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	11110	0000000010001000	0000000000001100
	Divisor Rotates right 1 bit.	11110	0000000001000100	0000000000001100
7	Remainder-Divisor<0	11110	0000000001000100	0000000000001100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	111100	0000000001000100	0000000000001100
	Divisor Rotates right 1 bit.	111100	0000000000100010	0000000000001100
8	Remainder-Divisor<0	111100	0000000000100010	0000000000001100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	1111000	0000000000100010	0000000000001100
	Divisor Rotates right 1 bit.	1111000	0000000000010001	0000000000001100
9	Remainder-Divisor<0	1111000	0000000000010001	0000000000001100
	Quotient Rotates left 1 bit, then setting 1 at bit0 of Quotient.	11110000	0000000000010001	0000000000001100
	Divisor Rotates right 1 bit.	11110000	0000000000001000	0000000000001100

Table 1. dividing 0Xffc by 0x11 or 111111111100 by 00010001



Quotient: F0

	Reg1: 0x133	Reg2:0x134
Remainder: F0	F	0

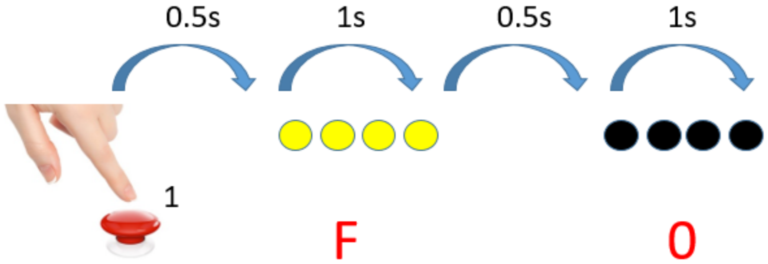
Remainder: 0C

	Reg1: 0x123	Reg2:0x124
Remainder: 0C	0	C

Figure 3

Quotient: F0

	Reg1: 0x133	Reg2:0x134
Remainder: F0	F	0



Remainder: 0C

	Reg1: 0x123	Reg2:0x124
Remainder: 0C	0	C

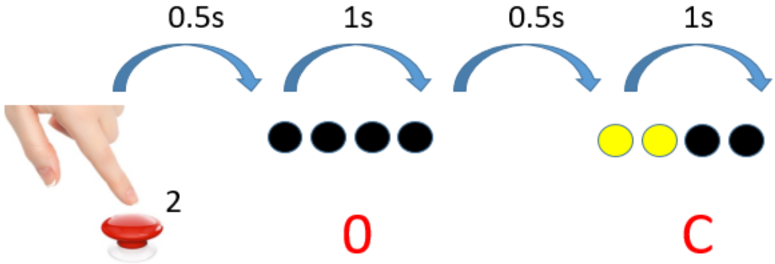


Figure4

Score: 100%

1. LED can be lighted up. (10%)
2. Setting lighting flag on ISR (Interrupt service routine). (10%)
3. Build Delay to control the interval time when Displaying digits time is 0.5 seconds and Lighting time is 1 second. (10%)
4. Explain how you design the delay macro to control the delay duration. (e.g. cycle used, input argument) (10%)
5. Quotient and Remainder are split into **two** 4-bit numbers head-to-head and the CORRECT results are shown on hardware. (20%)
   E.g, (A7) [1010 0111] > [1010] [0111]

	Bit7-bit4	Bit3-bit0
The address of Quotient	0x133	0x134
The address of Remainder	0x123	0x124

6. Be sure to use appropriate ADDRESSING MODE in your code. (Don't use MOVFF instruction in your whole code) (10%)
7. Pushing the button 1/button 2 will display digits of quotient/remainder sequentially. (from the bit7-bit4 to the bit3-bit0) (10%)
8. Display the CORRECT answer on software, after the division process is completed. (20%)

Restriction

1. Quotient on 0x133 and 0x134 and Remainder on 0x123 and 0x124
2. Lighting LED on RD3-RD0
3. Using direct (Banked) or indirect addressing mode in the 6th item of Score requirements.
4. Pull-up registers are required
5. Please put the value in the Specified address, TA can test your result quickly.

Hint:

1. Default Oscillator frequency: 4Mhz (set in the zipped sample code)
2. A little error on cycles and seconds are tolerated.
3. Easy questions: 1, 2, 3, 4, 6, 7