LEDs Shift Left and Right

黃永廣
助教：劉柏廷、郭柏興、林嘉偉、陳昶宏
雲科電子
2014/11/10

Reading Switches, Writing LEDs

- .ORIG x3000 ; file = “lab1a.asm”

- LD R1, Saddr ; R1 ←m[Saddr] = 0xFFFC = addr of SWs
- LDR R3, R1, #0 ; R3 ←m[0xFFFC+0] ; read switches

- LD R2, Laddr ; R2 ←m[Laddr] = 0xFFFD = addr of LEDs
- STR R3, R2, #0 ; m[0xFFFD+0] ← R3 = switches; write LEDs

- Saddr .FILL 0xFFFC ; Memory address for switches
- Laddr .FILL 0xFFFD ; Memory address for LEDs
- .END

Task1: LEDs Shift Left

- Input a bit pattern with switches SW7_0
- Show the pattern on LEDs, shift pattern left by one bit, then show pattern again
- Repeat showing on LEDs until pattern becomes 8 bits of 0, meaning all LEDs off
- Then Repeat reading from switches and doing the same steps again
- Can adjust pattern any time with switches
- Shift left by one bit is easy: N$\leftarrow$2N=N+N

Example

- SW[7:0]=b10011101 = 128+16+8+4+1=157
- N := SW[7:0] = b10011101 = 157
- LED[9:0] := N = b0010011101
- N := N+N = b0100111010 = 256+32+16+8+2 = 314
- LED[9:0] := N = b0100111010
- N := N+N = b1001110100 = 628
- LED[9:0] := N = b1001110100
- N := N+N = b0011101000
- …
- LED[9:0] = b0011101000→b0111010000→b1110100000
- →b1101000000→b1010000000→b0100000000
- →b1000000000→b0000000000
- Then repeat from the beginning

Task2: Shift LEDs Right

- One simple way to do right shift is to store each of the eight bits into eight different memory locations
- Each time, combine the eight words to form a 16-bit no., with the eight words corresponding to the rightmost 8 bits
- Write the 16-bit no. to the LEDs
- Then shift right: m[B0] $\leftarrow$ m[B1]; m[B1] $\leftarrow$ m[B2]; …
- Repeat the above forever

Right Shift of a Decimal Number

- E.g. N = #4329
- Digit0 = 9
- Digit1 = 2, NOT 20
- Digit2 = 3, NOT 300
- Digit3 = 4, NOT 4000
- Combine digits: N = 4*1000+3*100+2*10+9
- Shift right: Digit0=2, Digit1=3, Digit2=4, Digit3=0
- Combine digits: N= 0*1000+4*100+3*10+2=0432
- N: 4329→0432→0043→0004→0000→4329→…

Task 2a: Extract Each Bit from Switches E.g., SWs=b11110000=x00F0

- Getting each bit from pattern P and store each bit into memory as a word
- P=x00F0
- Mask ← x0001
- B0 ← P & Mask=x0000
- Mask ← Mask+Mask=x0002
- B1 ← P & Mask=x0000
- Mask ← Mask+Mask=x0004
- B2 ← P & Mask=x0000
- Mask ← Mask+Mask=x0008
- B3 ← P & Mask=x0000
- Mask ← Mask+Mask=x0010
- B4 ← P & Mask=x0010
- B4 ← x0001
- Mask ← Mask+Mask=x0020
- B5 ← P & Mask=x0020
- B5 ← x0001
- Mask ← Mask+Mask=x0040
- B6 ← P & Mask=x0040
- B6 ← x0001
- Mask ← Mask+Mask=x0080
- B7 ← P & Mask=x0080
- B7 ← x0001

Task 2b: Restore Number from Bits

- Given in memory
- B7 = 1
- B6 = 1
- B5 = 1
- B4 = 1
- B3 = 0
- B2 = 0
- B1 = 0
- B0 = 0
- Want to get: P = xF0
- $= 1*2^7+1*2^6+1*2^5+1*2^4 +0*2^3+0*2^2+0*2^1+0*2^0$

- P ← B7=b1=x1
- P ←2P+B6=b11=x3
- P ←2P+B5=b111=x7
- P ←2P+B4=b1111=xF
- P ←2P+B3=b11110=x1E
- P ←2P+B2=b111100=x3C
- P ←2P+B1=b1111000=x78
- P ←2P+B0=b11110000=xF0
- LEDs ← P

Task 2c: Shift Each Bit Right

- m[B0] ← m[B1]=0
- m[B1] ← m[B2]=0
- m[B2] ← m[B3]=0
- m[B3] ← m[B4]=1
- m[B4] ← m[B5]=1
- m[B5] ← m[B6]=1
- m[B6] ← m[B7]=1
- m[B7] ← 0
- Ready to do Task 2b again

- Initial P=0xF0
- Form P=x78, x3C, x1E, x0F,
- x07, x03, x01, x00,
- Read switches again…

Task 2 by Doing Tasks 2a, 2b, 2c

- Task 2a: Extract each bit of the number read from switches and store into memory
- Task 2b: Read each bit from memory, restore the number from the 8 bits and show on LEDs
- If the no. is 0, then jump to Task 2a, else Task 2c
- Task 2c: Read each bit from memory and store it to the memory location to its right
- Jump to Task 2b

Algorithm

- Start: Pattern ← SW7_0;
- Mask ← 0x0001;
- B0 ← Pattern & Mask
- Mask ← Mask*2=0x0002;
- B1 ← Pattern & Mask
- If (B1!=0), B1 ←1;
- ….

- Loop: P ← B7;
- P ← 2P+B6;
- P ← 2P+B5;
- …
- P ←2P+B0;
- Leds ←P;
- B0 ←B1; B1 ←B2;
- ….;
- If P==0 Goto Start
- Else Goto Loop

Program Framework

- .ORIG x3000
- …
- Bye BRnzp Bye

- B7 .FILL x0 ; Bit 7
- B6 .FILL x0 ; Bit 6
- B5 .FILL x0 ; Bit 5
- B4 .FILL x0 ; Bit 4
- B3 .FILL x0 ; Bit 3
- B2 .FILL x0 ; Bit 2
- Bit1 .FILL x0 ; Bit 1
- Bit0 .FILL x0 ; Bit 0
- .END

Task 3

- If in task 2, your program is very long because you did not use loop,
- Then in task 3, you should use loops to make your program much shorter
- If in task 2, you already use loops then jump to task 4 in the next slide

Task 4

- Write a program to make LEDS shift right with the idea of division by 2

Task 5

- Write a program to make LEDs shift right with an algorithm different from those of task 2, task 3, and task 4