CSC 258 after midterm

JoJo

进度review

Mar 6 – 10	Architecture & microprogramming	Lab 6
Mar 13 – 17	Assembly language basics	Lab 7 & Project proposal
Mar 20 – 24	Assembly language program design	Project demo #1
Mar 27 – 31	Advanced assembly language	Project demo #2
Apr 3 – 5	Topic overflow & course review	Project demo #3 & project report

- ALU – process data
- Storage - Store data
- Control unit – FSM to control the process

LAB 6

- FSM
- $Ax^2 + Bx + C$

Review Before midterm…

Combinational Logic [What to do]

- Used for Computation
  - Adder, Comparator, ALU
- For relation and connection
  - In FSM design etc…

Time Sequential Logic [when to do]

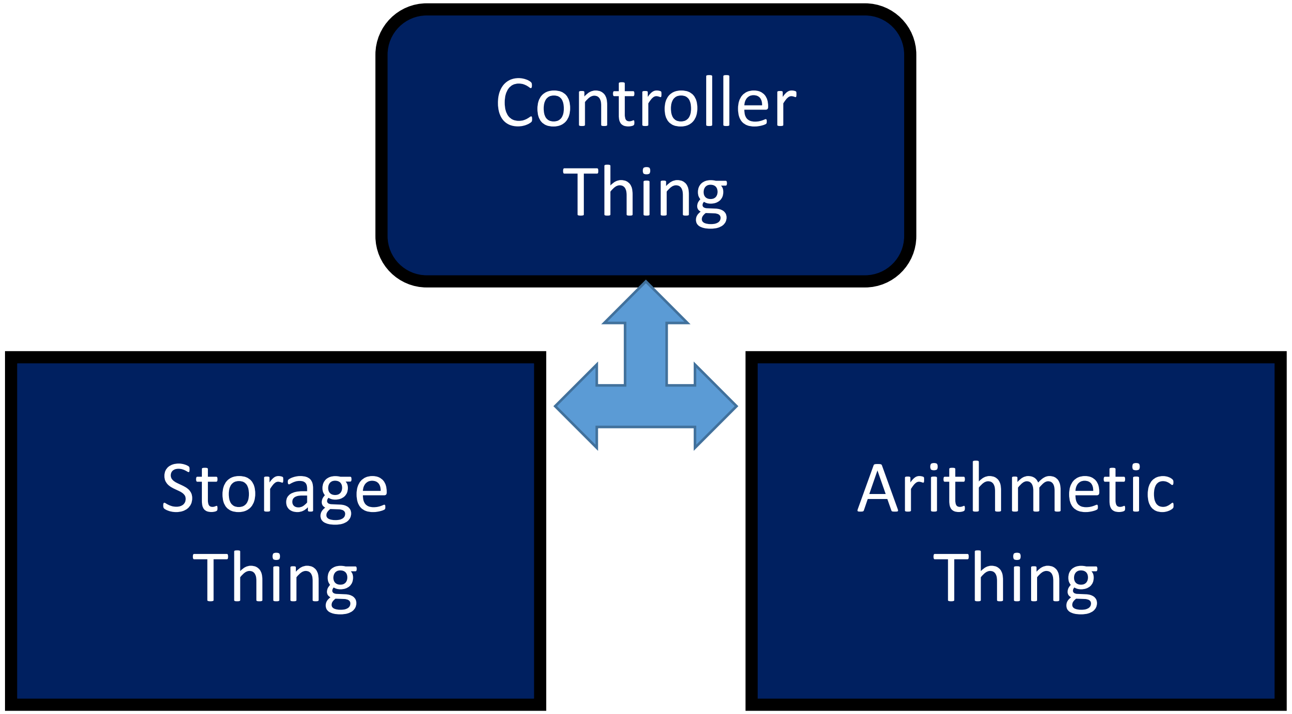
- Used for storing information
  - Shift/load register
- For designs related with time
  - Counter
  - FSM

Microprocessor

- Process data
- Store data
- Control

Microprocessor

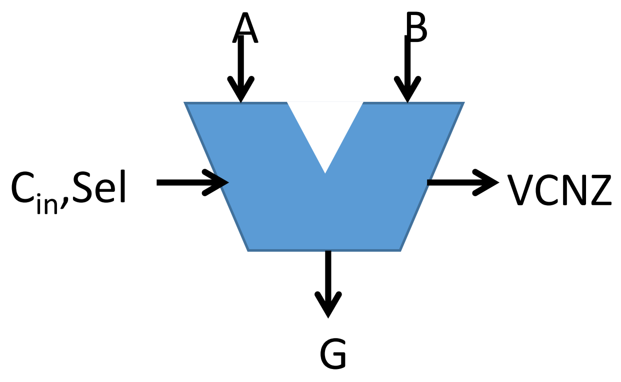
- ***Defn***: A **microprocessor** is a computer processor which incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC), or at most a few integrated circuits.



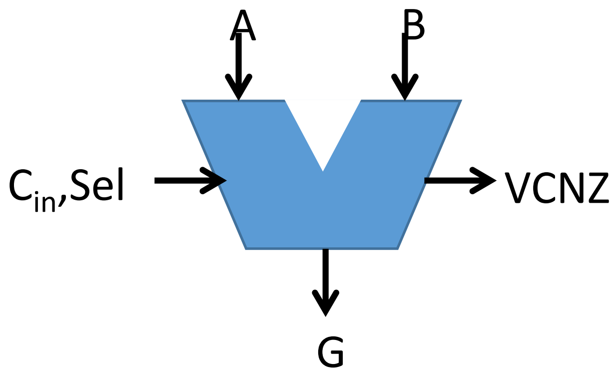
ALU(Arithmetic Logic Unit)

- What we have before: (一些分立的计算器)
  - Basic Logic computation : AND/OR/NOT/… GATEs
  - Basic math computation : adder/substractor (multiplication and divide ???)
  - Basic comparison
- What we want: (一个完整的计算器)
  - A integrated Computation Block that can handle all

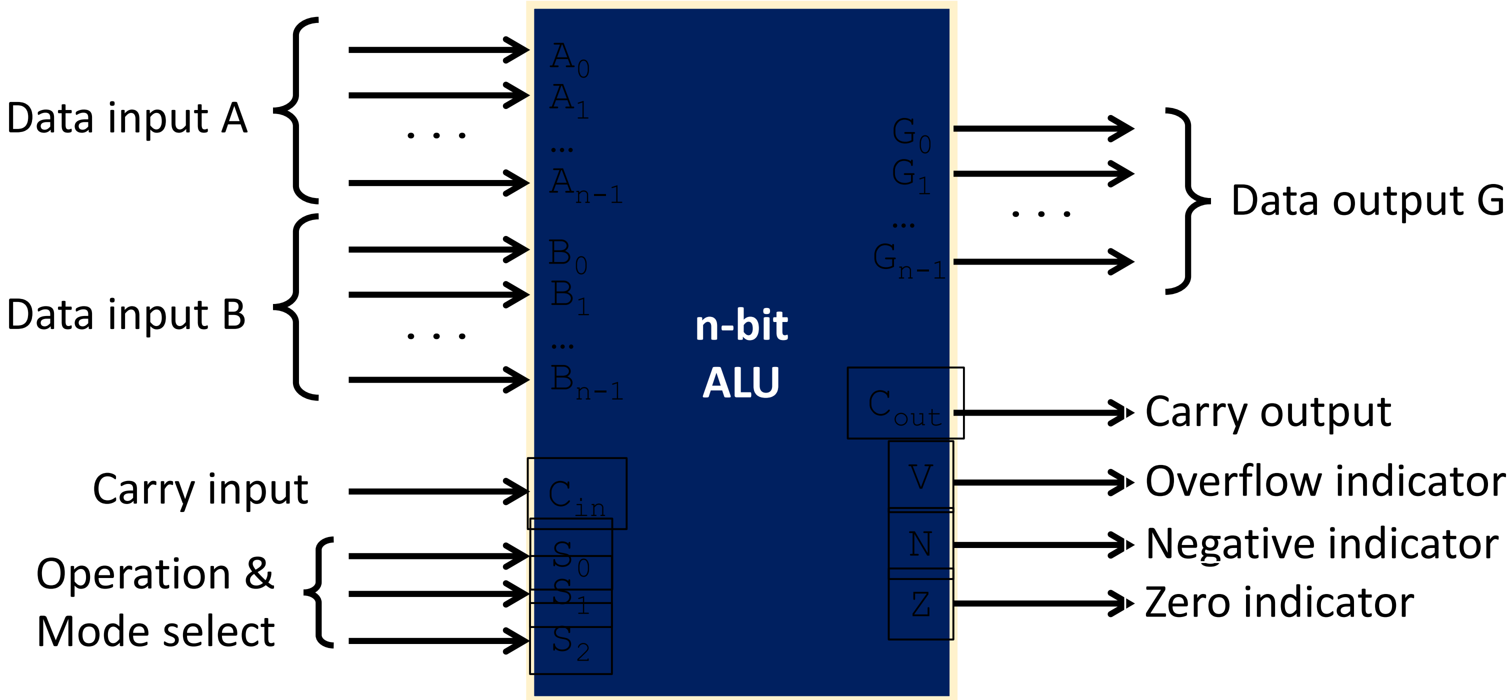
- What we have before:
  - module and(a, b, y)
  - module or(a, b, y)
  - module adder(c_in, a, b, c_out)
- What we want:
  - module (A, B, C_in, Sel, V, C, N, Z, G)
    - ……



ALU(Arithmetic Logic Unit)



- **V**: o**V**erflow condition
  - The result of the operation could not be stored in the n bits of G, meaning that the result is incorrect.
- **C**: **C**arry-out bit
- **N**: **N**egative indicator
- **Z**: **Z**ero-condition indicator



ALU(Arithmetic Logic Unit)

Select		Input	Operation	
S_1	S_0	Y	$C_{in}=0$	$C_{in}=1$
0	0	All 0s	G = A (transfer)	G = A+1 (increment)
0	1	B	G = A+B (add)	G = A+B+1
1	0	B	G = A+B	G = A+B+1 (subtract)
1	1	All 1s	G = A-1 (decrement)	G = A (transfer)

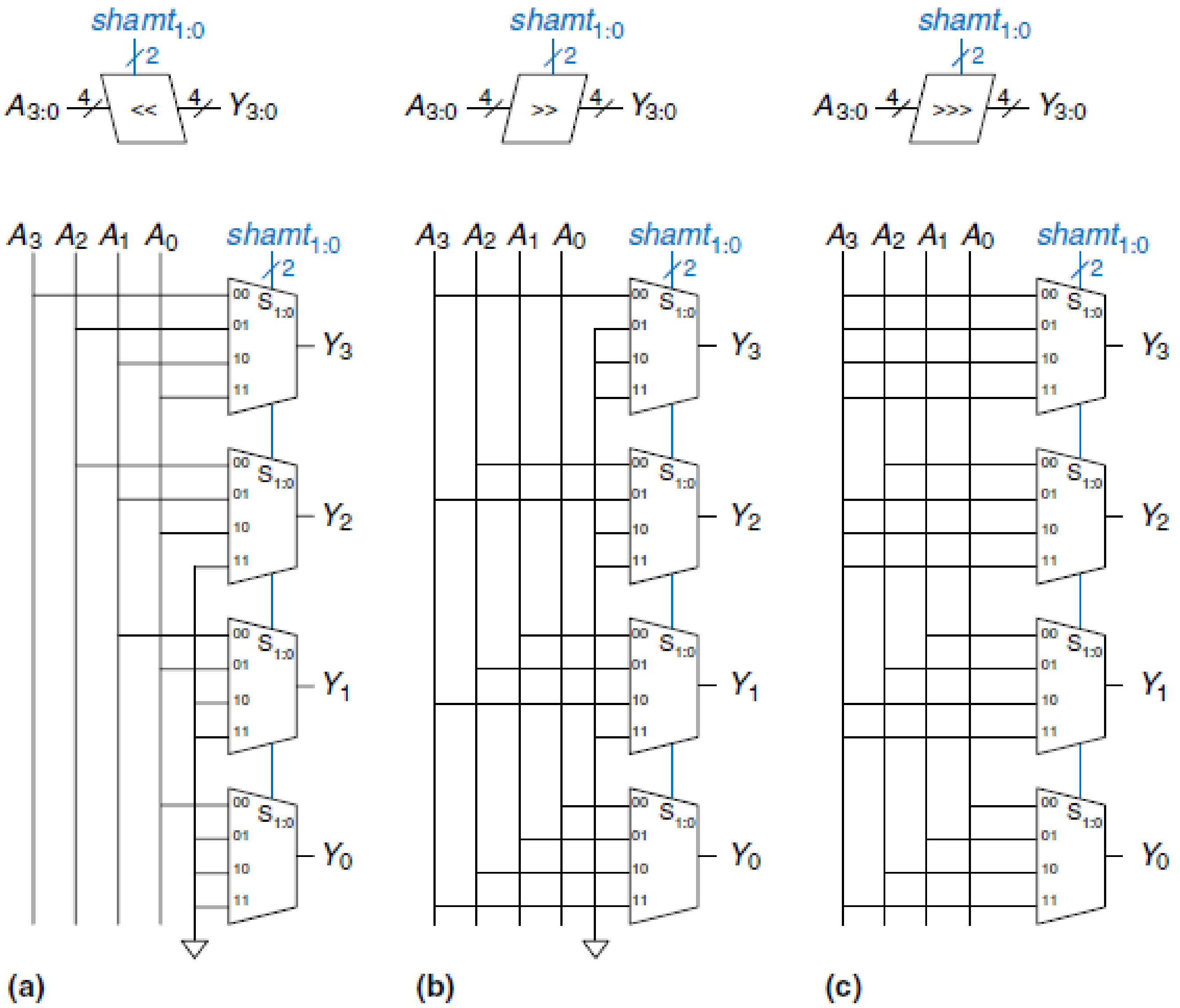
- If S2 = 1, then logic circuit block is activated, the combination of S1 and S0 can be use SEL signal to choose logic computation like AND, OR, NOR, NOT (up to 4)
- What device should we use here to determine which computation goes to output? A, MUX; B, FSM; C, Decoder; D, Sorting Hat

Shifters and Rotators

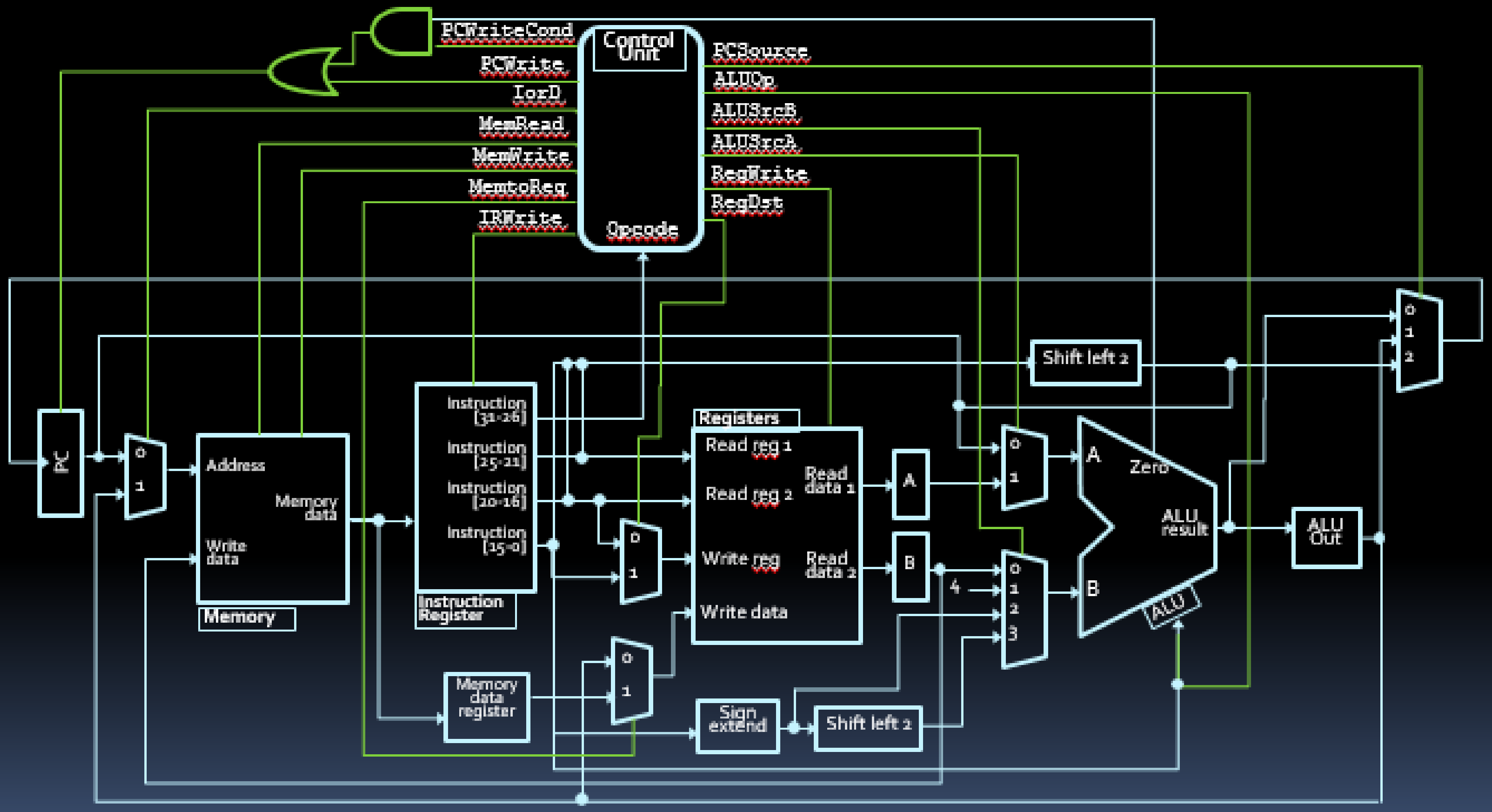
- ***Shifters*** and ***rotators*** move bits and multiply or divide by powers of 2.
- **Logical shifter** — shifts the number to the *left (LSL) or right (LSR)* and fills empty spots with 0's.
  - Ex: **11001** LSR 2 **00110**;   **11001** LSL 2 **00100**
- **Arithmetic shifter**—is the same as a logical shifter, but on right shifts fills the most significant bits with a copy of the old most significant bit (msb).
  - Ex: **11001** ASR 2 **11110;**   **11001** ASL 2 **00100**
- **Rotator**—rotates number in circle such that empty spots are filled with bits shifted off the other end.
  - Ex: **11001** ROR 2 **01110**; **11001** ROL 2 **00111**
- **Special Case(Multiplication) :** 000011(2) << 4 = 110000(2) ⇔ 3(10)*2^4 = 48(10)
- **Division: shift to ________________ ?**

Shifters and Rotators

- shamt[1:0] means **sh**ift **amount**
- (a) shift left (<<)
- (b) logical shift right (>>)
- (c) arithmetic shift right (>>>)
- 思考题：draw a 4 bit rotator (left / right)



The Final Destination

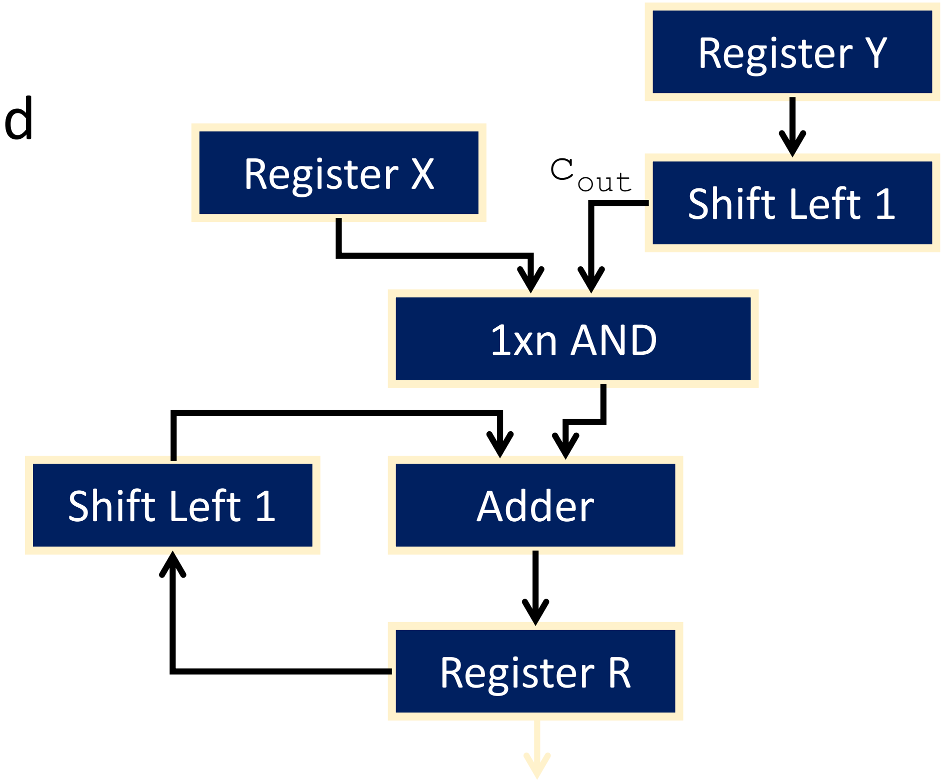


Multiplication (accumulator)

- We want to implement multiplication by **loop.**
- In (i_th) loop
  - if Y(i) = 1, we will shift (i - 1) bit X and add the previous result.

230	multiplicand		0101
× 42	multiplier		× 0111
460	partial		0101
+ 920	products		0101
9660			0101
			+ 0000
	result		0100011

$2\ 30 \times 42 = 9660$ $\qquad$ $5 \times 7 = 35$



Example

- $M * 0011\ 1110 = M*(2^5 + 2^4 + 2^3 + 2^2 + 2^1) = M * 62$
- Complexity: Shift 5 times and add 5 times;
- If $M*(2^{10} – 1)$
  - Shift ? times and add ? times

- How to reduce the algorithm complexity?
  - $M * 9999 = M*(9*10^4 + 9*10^3 + 9*10^2 + 9*10^1) = M*(10000 - 1)$

    $= M*10^5 – M*1$

Booth algorithm

- 将乘法（分别按位求和，add次数：N-1）根据（01）和（10）的位置转换为纯粹的shift 以及 少量的求和（add次数：n）
- M * 0011 1110 = M * (0100 0000 – 0000 0010)

  = M*(2^6 – 2^1) = M*62
- 事实上，任何二进制数中连续的1可以被分解为两个二进制数之差：
- 0000 1111 0000 = 0001 0000 0000 – 0000 0001 0000;
- 因此，我们可以用更简单的运算来替换原数中连续为1的数字的乘法，通过加上乘数，对部分积进行移位运算，最后再将之从乘数中减去

Example

- M * 00111010 = M*(2^5 + 2^4 + 2^3 +2^1) = M*58
  = M*(2^6 – 2^3 + 2^1)

- Booth Algorithm在遇到一串数字中的第一组从0到1的变化时（即遇到01时）执行加法
- 在遇到这一串连续1的尾部时（即遇到10时）执行减法。
- 这在乘数为负时同样有效。当乘数中的连续1比较多时（形成比较长的1串时），布斯算法较一般的乘法算法执行的加减法运算少。

Step by Step

- 例子参考Example : (-5) * 2 = (-10)
  - (1) A = 11011, B = 00010, P = 00000 00000,
  - (2) Add an extra zero bit to the right-most side of A: A = 110110

  - Repeat length(original (A)) times:
    - Last 2 digits of A = '10' → subtract B from MSB of P → P = 11110 00000 and Arithmetic shift P and A one bit to the right: A = 111011, P = 11111 00000
    - Last 2 digits of A = '11' or '00' → do nothing and Arithmetic shift P and A one bit to the right: A = 111101, P = 11111 10000
    - Last 2 digits of A = '01' → add B to the MSB of P → P = 00001 10000, and Arithmetic shift P and A one bit to the right: A = 111110, P = 00000 11000
    - …… keep repeating until reach length of original bit long of A = 5;
    - Get final P = 11111 10110 = -10

12fall final考题(by **Steve Engels**)

- In the space below, perform Booth's Algorithm on the binary values A=10110 and B=01101. Show your steps in the space provided. **(6 marks)**
- **Solution:**
- **Step (0)** P = 00000 00000, A = 10110 0, -B = 10011
- Entering the loop (for i = [1 : length(original(A))])
- **Step (i = 1):** Last 2 digits of A = '11' or '00' → do nothing and Arithmetic shift P and A one bit to the right: A = 110110, P = 00000 00000
- **Step (i = 2):** Last 2 digits of A = '10' → subtract B from MSB of P → P = 10011 00000 and Arithmetic shift P and A one bit to the right: → P = 11001 10000 , A = 111011;
- **Step (i = 3):** similar with i = 1 case, Arithmetic shift P and A one bit to the right: A = 111101, P = 11100 11000;
- **Step (i = 4):** Last 2 digits of A = '01' → add B to the MSB of P → P = 01001 11000, and Arithmetic shift P and A one bit to the right: A = 111110, P = 00100 11100;
- **Step (i = 5):** same with (i = 2), subtract → P = 10111 11100 and arithmetic shift to right: → **P = 11011 11110**
- **Final P = 11011 11110.**