Class 6:
Intro to SRC Simulator
Register Transfers and Logic Circuits

SRC Language Conventions

- Chapter 2, Appendix B.5

SRC Simulator Demo

- cond_br.asm (SRC)
- fib_simple.asm (SRC)
  - Iterative fibonacci using only registers
- array accesses (pseudo-Sparc)

cond_br.asm

```
#define Cost 125
if(X<0)
  X = -X;

// Example program
// +6 in register -> false
// - 6 in register -> true
```


fib_simple.asm (C style source)

```
register int current_result = 0;      // r1
register int fib_n_1 = 1;             // r2 – fib(n-1)
register int fib_n_2 = 0;             // r3 – fib(n-2)
register int count = 5;               // r4 – loop ctr

while(count != 0)
{
  current_result = fib_n_1 + fib_n_2;
  fib_n_2 = fib_n_1;
  fib_n_1 = current_result;
}
```


Arrays in C

```
int    a[5];
int main()
{
       int i=0;
       for(i=0; i<5; i++)
              a[i]=i;
       return 0;
}
```

Arrays in Assembly

```
// Array access example (pseudo-Sparc assembly
   code)

 mov 0, r2          ; i in r2
 mov a, r3          ; r3 = address of a
Label1:             ; body of loop
 sll r2, 2, r4      ; mult by 4)
                    ; r4=i*4, offset calculation
 st r2, [r3+r4]     ; store i into a + 4*i
 add r2, 1, r2      ; add 1 to i
 cmp r2, 4          ; compare i to 4
                    ; loop will iterate 5 times
 ble Label1         ; branch less than equal to
                    ; Label1
 stop
```

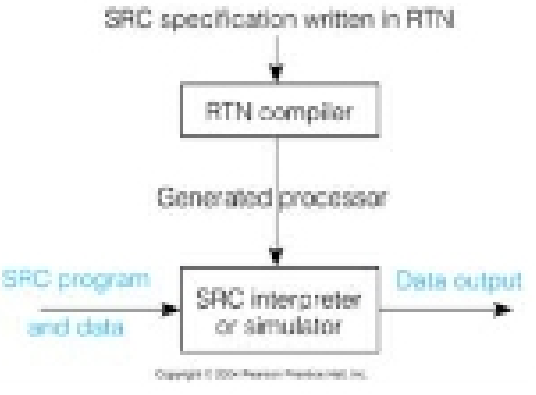

Register Transfer Notation

- Formal description language
  - *Metalanguage*
    - Language used to describe languages
    - Used to describe SRC machine
  - Mathematical notation

Why are Specification Languages Useful?

- **Allows:**
  - Description of **what** without having to specify ***how***.
  - **Precise and unambiguous** specifications, unlike natural language.
- **Reduces errors:**
  - Due to **misinterpretation of imprecise specifications** written in natural language
  - Due to **confusion in design and implementation** - "human error."
- **Specifications can be automatically checked and processed by tools.**
  - An RTN specification could be input to a simulator generator that would **produce a simulator** for the specified machine.
  - An RTN specification could be input to a compiler generator that would **generate a compiler** for the language, whose output could be run on the simulator.

Relationship of RTN to SRC



Road Map

How do data transfer operations map to logic circuits?

Assembly language

Architecture

Gate-level logic

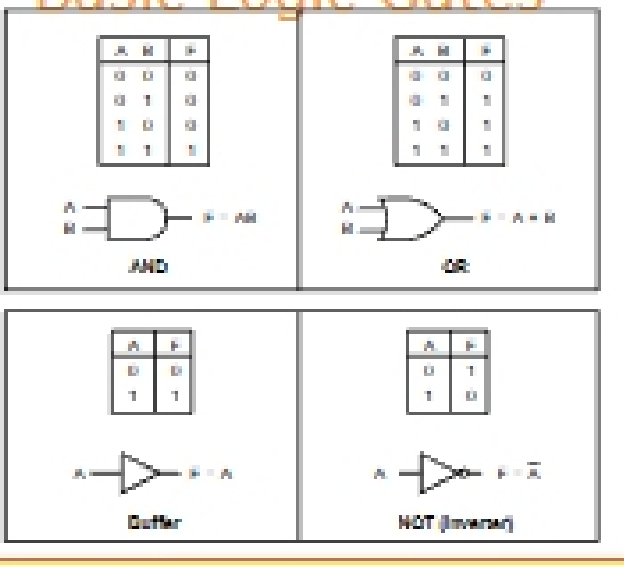
DLD Review

Appendix A

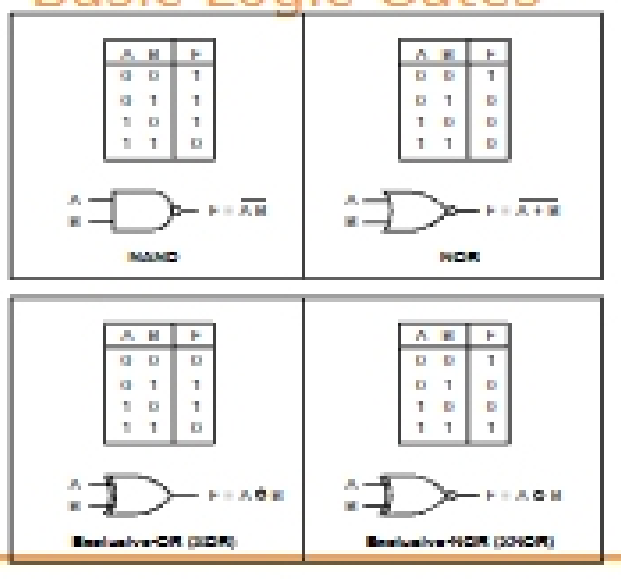
Definitions

- Combinational logic
  - a digital logic circuit in which *logical decisions* are made based only on *combinations* of the inputs. e.g. an adder.
- Sequential logic
  - a circuit in which decisions are made based on combinations of the current inputs as well as the past history of inputs. e.g. a memory unit.
- Finite state machine
  - a circuit which has an *internal state*, and whose outputs are functions of both current inputs and its internal state. e.g. a vending machine controller.

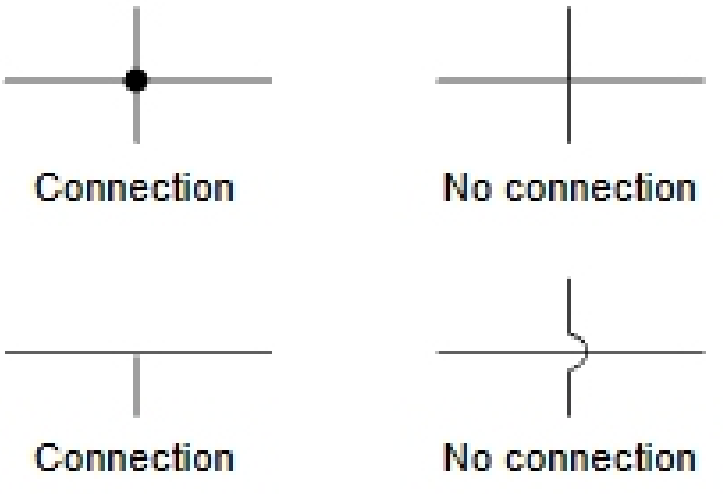
Basic Logic Gates



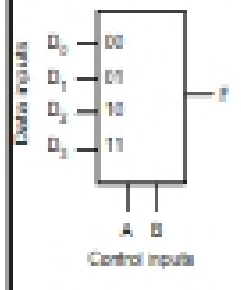
Basic Logic Gates



Circuit Intersections



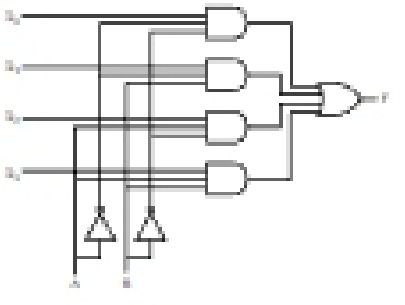
Multiplexer (MUX)



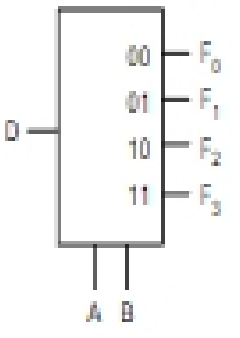
A	B	F
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3

Inputs: several
Outputs: one

$F = \bar{A}\bar{B}D_0 + \bar{A}BD_1 + A\bar{B}D_2 + ABD_3$



Demultiplexer (DEMUX)



D	A	B	F_0	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Input: 1
Output: several possible

$F_0 = D\bar{A}\bar{B}$ $F_2 = DA\bar{B}$

$F_1 = D\bar{A}B$ $F_3 = DAB$