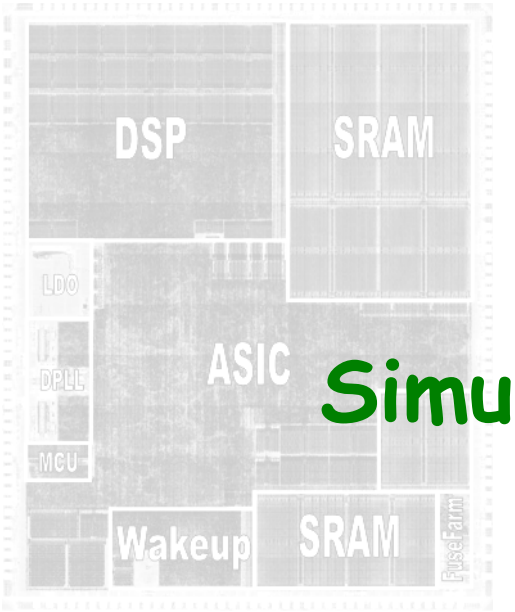
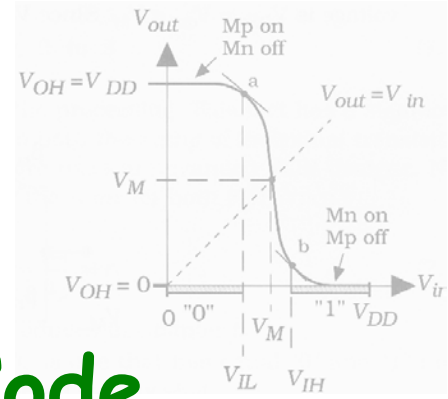
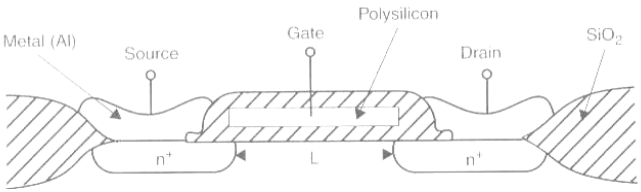


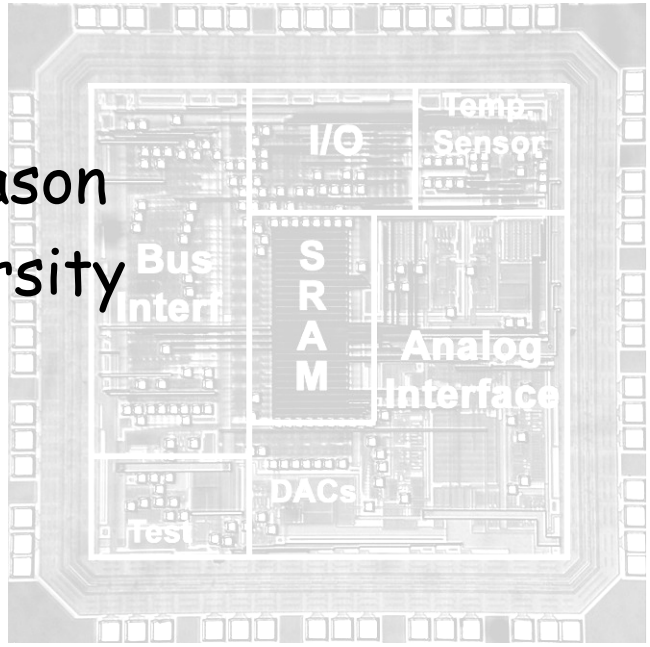
$$(a+b)\cdot(a+c) = a+a\cdot b+a\cdot c+b\cdot c$$
$$= a\cdot(1+b)+a\cdot c+b\cdot c$$
$$= a\cdot(1+c)+b\cdot c$$
$$= a+b\cdot c$$



ECE 331: PC Lab 2
Simulating Simple ASM Code

Professor Andrew Mason
Michigan State University
Spring 2013





Outline

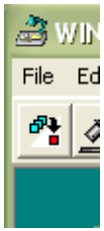
- Announcements
  - HW6 due Mon
  - Lab 4 next week
  - Quiz 2 next Wednesday
  - Midterm Exam in 2 weeks; no lab week of exam

This lab uses files:
PCLab2-4.asm
PCLab2-5.asm

- Objectives
  - Write short ASM instruction blocks to achieve specific program tasks
- Topics
  - Assembly Process
  - Review starting and using ASM development environment
    - save, compile, simulate, step/trace, observe
  - Simple loop
  - List-copy loop with indexed addressing
  - Loop with BRCLR/BRSET
  - Writing and debugging loop program

Using WinIDE Dev. Environment

- Launch application
  - START > All Programs > P&E… > WinIDE Development Environment
- Use editor: File > New File
  - type code
  - save file (e.g., ece331/ex2.1.asm)
- Assemble
  - click on Assemble/Compile File icon



These are not instructions for you to follow; it is a summary of how to use the simulator.

- Check for errors
  - look for error message at bottom of window
  - view .lst file for info on errors
- Simulate
  - click on Simulator (EXE2) icon

  - set Program Counter (PC) to program starting address
    - left-click on PC in CPU12 Window or type PC 4000 in command line
  - view source code disassembled
  - simulate: click on Go! icon

Exercise 2-1: Starting off easy…

1. Write a program that will
   - load #$0F into accA
   - AND accA with #$F0
   - store results to [$5550]
2. Save program as ex2-1.asm
3. Compile, simulate
   - fix syntax errors
   - set PC 4000 in simulator
4. Verify result
   - look in memory for expected result
   - Is correct value in proper location at end of program?

Program & Data Memory
When downloading code to a hardware device, where you store program and data bytes depends on what type of memory is at which addresses. When simulating, it really does not matter.
Let's assume
Program starts at $4000
Data contained within $2000-2FFF

Something to get you started

```
;331 Exercise 2-1 –Load and logic
        ORG     $4000   ;top of code
        LDAA    #$0F
        ANDA    #$F0
        STAA    $5550
        END
```

Exercise 2-2: Simple Modifications

1. Change initial value in accA
   - compile, simulate, verify result
   - Is expected value stored in proper location?

2. Save program as new name (e.g., ex2-2.asm)
3. Change Load and AND instructions to use *extended* rather than *immediate* address modes
   - must use FCB directive to store data to memory
     - see note on previous slide: start data at $2000
   - compile, simulate, verify result
   - Is expected value stored in proper location?

Example directives to help with this task

```
ORG     $2000   ;top of data
FCB     $0F,$F0
```

Exercise 2-3: Using Index Registers

1. Save program as new name (e.g., ex2-3.asm)
2. Change Store to use indexed address mode
   - must load *reference* address into a register (IX or IY)
   - compile, simulate, verify result
   - Is expected value stored in proper location?

Examples of instructions to will need to add
(not correct instructions, just format examples)

```
LDX    #$2000
STX    $02,X
```

Exercise 2-4: Simple looping example

1. Download "PCLab2-4.asm" from class website
2. Edit program to make it work
   - should initialize 10 data bytes and a SUM, then sum the 10 values
3. Compile, simulate
4. Debug and verify
   - remember the step and breakpoint functions
5. Modify code so that counter starts at 10 ($0A) and counts down to 0.
   - what lines have to be changed?

Question:

- How would you do this task without indexed addressing mode?

Exercise 2-5: CLR/SET Bit & Branch

- Goal: Write a program that will
  - Read through a data memory block
  - If value is an odd number (determined by BRSET or BRCLR)
    - end
  - Else
    - set bits 0 and 1 to '1'
    - set bits 2 and 3 to '0'
    - load result to accB
    - read next value
- Download "PCLab2-5.asm" from class website
  - code has 5 errors → fix them
- When running correctly
  - should see accA=06 and accB=03 at end of program
- Turn in properly functioning ASM code with Homework 6