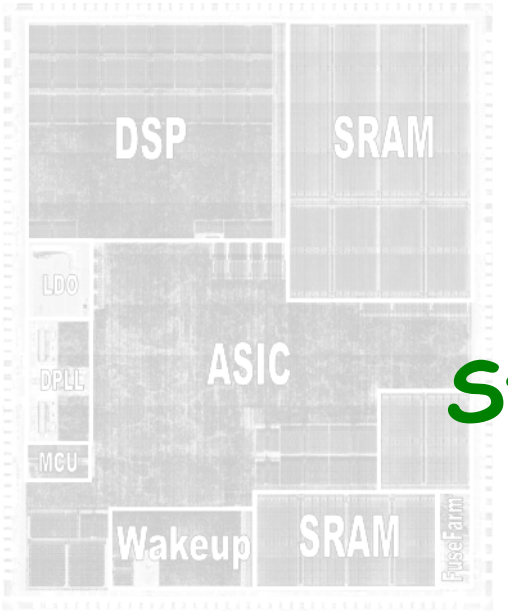
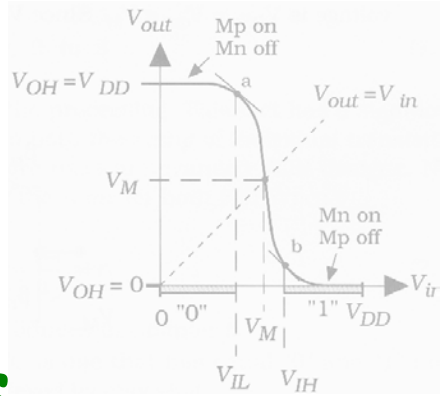
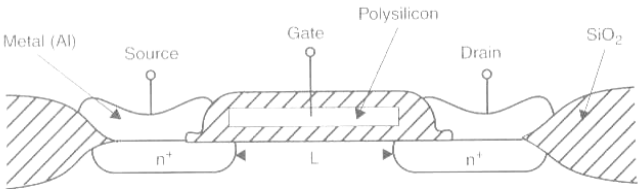


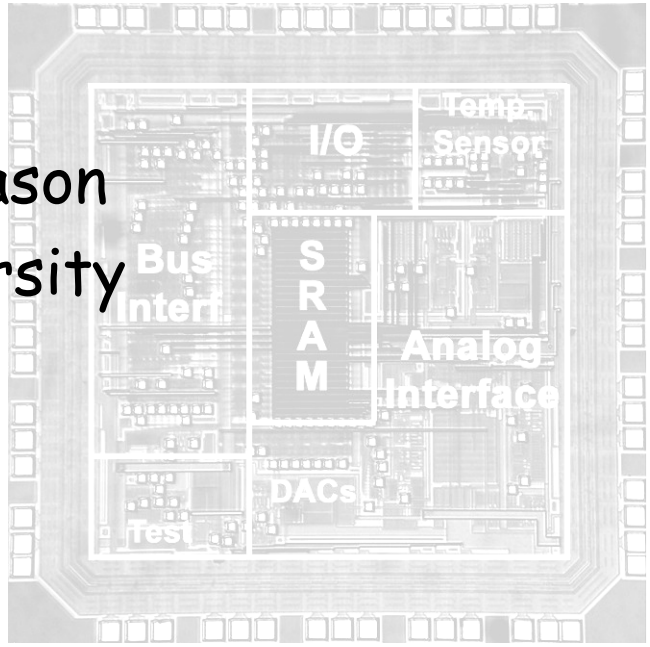
$$(a+b)\cdot(a+c) = a+a\cdot b+a\cdot c+b\cdot c$$
$$= a\cdot(1+b)+a\cdot c+b\cdot c$$
$$= a\cdot(1+c)+b\cdot c$$
$$= a+b\cdot c$$



ECE 331: PC Lab 3
Stack and Subroutines

Professor Andrew Mason
Michigan State University
Rev: S11



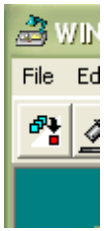


Outline

- Announcements
- Objectives
- Topics
  - Review starting and using ASM development environment
  - Pushing data to the stack
  - Pulling data from the stack
  - Calling subroutines
  - Simple subroutine 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/ex1.asm)
- Assemble
  - click on Assemble/Compile File icon



- 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 1: Push it

Print the check-off sheet on slide 8

1. Create this program →

```
;PClab3_ex1
            ORG     $4000
            LDS     #$6000
            LDAA    #$AA
            LDAB    #$BB
            LDX     #$0F0F
            LDY     #$5533
            SWI
            END
```

2. Add lines to the program to **push** accA, accB, iX, & iY to the stack, in that order
3. Compile & remove syntax errors
4. Start simulator
   - set PC = 4000
   - display code at 4000
   - observe memory at 5FF0 - 6010
5. Trace through program and observe stack being filled
6. Fill in the check-off sheet when program stops executing

Exercise 2: Pull it

Do not begin until Exercise 1 is complete

1. Save a copy of the Exercise 1 program as Exercise 2 (e.g., PClab3_ex2)
2. At the end of the program add instructions to **pull** accA, accB, iX, & iY from the stack, in that order
3. Compile & remove syntax errors
4. Start simulator
   - set PC = 4000
   - display code at 4000
   - observe memory at 5FF0 - 6010
5. Test program to ensure it is working as expected
   - use trace and break-points as necessary
6. Fill in the check-off sheet when program stops executing

Exercise 3: Pull it in subroutine

Do not begin until Exercise 2 is complete

1. Save a copy of the Exercise 2 program as Exercise 3
2. Modify the program so that:
   - the pull instructions are in a subroutine beginning at $4400
   - the main program calls the subroutine before the SWI
3. Compile & remove syntax errors
4. Start simulator
   - set PC = 4000 - display code at 4000
   - observe memory at 5FF0 - 6010
5. Test program to see if it is working as expected
   - use trace and break-points as necessary
   - if you have entered the program as instructed, it should not work
6. Track down the problem and fix it!
   - also modify the pulls so they properly restore registers
7. Complete the check-off sheet for Exercise 3

```
;program structure
4000    Main program
        -
        -
        JSR
        SWI
4400    Subroutine
        -
        RTS
        END
```

Exercise 4: Subroutine

1. Write a program that will
   - load $DDDD into iX and $4444 into iY
   - load $05 into accA and $50 into accB
   - jump to a subroutine "sum" that
     - saves contents of iX and iY on stack & restores at end of subroutine
     - load $0101 into iX and $2020 into iY
     - adds the values in accA and accB and stores the sum in accA
     - sets accB = FF if the sum has carry overflow; otherwise accB = 00
   - loads $8E into accA and $E8 into accB
   - jumps to subroutine "sum"
   - end (SWI, END)
2. Compile & remove syntax errors
3. Simulate program to ensure it is working correctly
4. Complete the check-off sheet for Exercise 4

PC Lab 3 Check-off Sheet

Student Name: _____

Exercise 1

Record the register values observed from program simulation at the end of Exercise 1.

Acc A	*Acc B*	*Index X*	*Index Y*	*Stack Pointer*

Record the stack contents observed from program simulation at the end of Exercise 1.

Addr.							
Value							

Exercise 2

Record the register values observed from program simulation at the end of Exercise 2.

Acc A	*Acc B*	*Index X*	*Index Y*	*Stack Pointer*

Is this what you expect? If not, explain.

_____

_____

_____

_____

Exercise 3

What changes were necessary for the program to work correctly?

_____

_____

_____

_____

Record the register values observed from program simulation at the end of Exercise 3.

Acc A	*Acc B*	*Index X*	*Index Y*	*Stack Pointer*

Exercise 4

Record the register values observed from program simulation at the end of Exercise 4.

Acc A	*Acc B*	*Index X*	*Index Y*	*Stack Pointer*