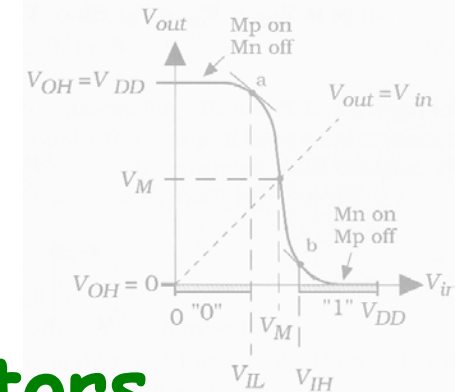


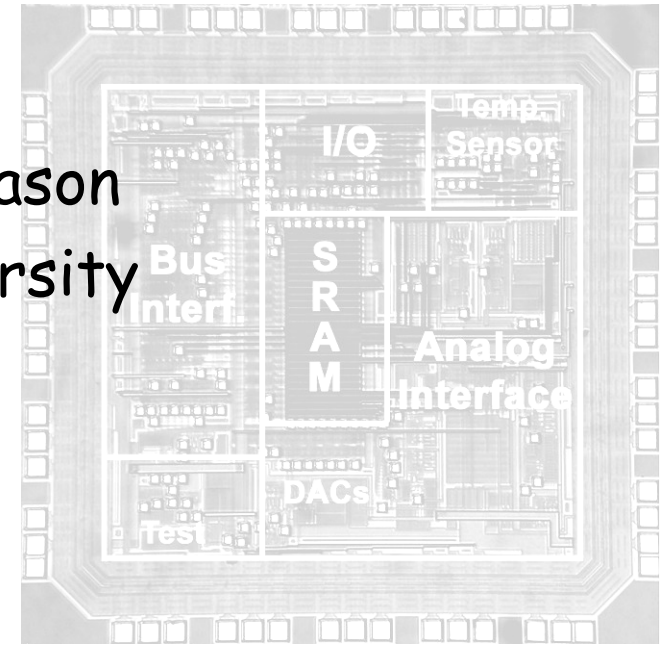
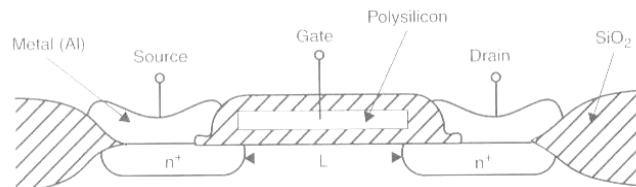
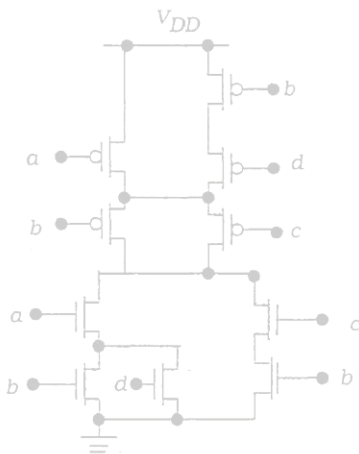


$$\begin{aligned}
 (a+b) \cdot (a+c) &= a + a \cdot b + a \cdot c + b \cdot c \\
 &= a \cdot (1+b) + a \cdot c + b \cdot c \\
 &= a \cdot (1+c) + b \cdot c \\
 &= a + b \cdot c
 \end{aligned}$$



ECE 331: PC Lab 1: Using HC12 ASM Simulators

Professor Andrew Mason
Michigan State University
Spring 2012



Outline

- Announcements
 - HW5 due next Wednesday
 - Quiz 2 in two weeks (Wed Feb 22)
- Objectives
 - List and identify ASM directives
 - Use ASM simulator program to test and debug HC12 ASM code
 - Write short ASM instruction blocks to achieve specific program tasks
- Topics
 - ASM Directives described
 - Development environments
 - Opening ASM simulator; typing and saving ASM code
 - Assembling programs
 - Observing simulation results of ASM directives
 - Observing results of example ASM code blocks
 - Stepping and tracing program execution; using break points
 - Tracing code block with simple branch (BRA)



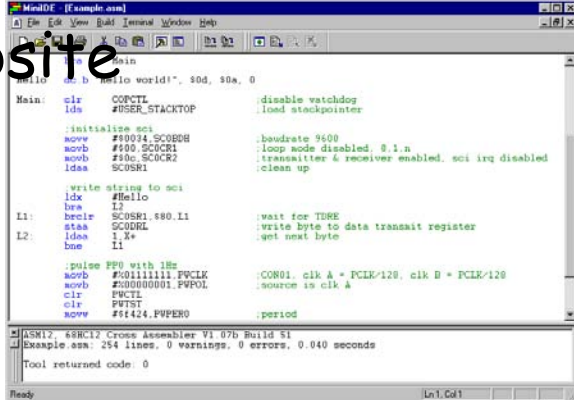
ASM Directives

- See Handout 4
- Critical Directives
 - **ORG** → defines starting memory location of ASM code
 - **END** → defines end of program
- Helpful Directives
 - **SWI** → set software interrupt; program will halt here
 - **EQU** → assigns text string to hex value
- Data Storage Directives
 - **FCB** → form constant byte (write value to memory)
 - FDB, 2 byte version of FCB
 - **RMB** → reserve memory byte(s), hold/identify block of memory
- Other
 - **FCC** → stores ASCII code of text characters in memory



HC12 ASM Development Environments

- Development Environment
 - software tool containing editor, assembler (assembly compiler), simulator, and/or serial communication to a development board
- Development Board
 - hardware PCB containing a microcontroller, memory, and I/O devices used to test and develop microcontroller code
- Textbook Simulator
 - on CD in back of textbook and on EGR PCs (P&E ... HC12...)
 - all-in-one editor, assembler, simulator
- Freeware simulators linked on class website
 - MGTEK MiniIDE
 - color coded text editor, assembler
 - 68HC12 Simulator (Java)
 - no built-in text editor, simulator only
 - loads (compiled) S19 files, decent trace/breakpoint features



```
MiniIDE [Example.asm]
[File Edit View Build Internal Window Help]
Main:  clr COPCTL           ;disable watchdog
      lds #USER_STACKTOP ;load stackpointer
      ;initialize sci
      move #9604,SC0B0H   ;baudrate 9600
      movb #90,SC0CR1    ;loop mode disabled 0.1.n
      movb #90,SC0CR2    ;transmitter & receiver enabled, sci irq disabled
      ldsa SC0SR1        ;clean up
      ;write string to sci
      ldx #Hello
      bea l2
      brair SC0SR1,$0.L1 ;wait for TDR
      stas SC0DR1        ;write byte to data transmit register
l2:   ldsa l2,*
      bne l1
      ;pulse PPS with 1Hz
      movb #00000001,PCLK ;CON01, clk A = PCLK/128, clk B = PCLK/128
      movb #00000001,PFPOL ;source is clk A
      clr PUCT1
      clr PUST1
      movb #61424,PUPER0   ;period

ASM1: s8HC12 Cross Assembler V1.07b Build 51
Example.asm: 254 lines, 0 warnings, 0 errors, 0.040 seconds
Tool returned code: 0
```



Starting WinIDE Dev. Environment

- Launch application
 - START > All Programs > P&E... > WinIDE Development Environment
- Setup -none needed, can play with options later
- Use editor: File > New File
 - Example: Using directives

;331 Example 1 -Directives

;**begin data block definition

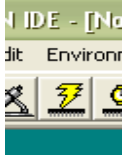
```
TDR    EQU    $6000    ;top of data record
        ORG    TDR      ;set origin
        FCB    $AA,$BB,$CC,$FF,$EE,$01
        SWI
        END
```

use tabs and spaces
to organize your code
and make it easy to read

- Save file (ece331/ex1.asm)
- Assemble
 - click on Assemble/Compile File icon
- Check for errors
 - look for error message at bottom of window
 - view .lst file created in directory with .asm file (use Notepad or similar)



Assembling & Simulating ASM Code

- Load assembled code (.s19 file)
 - not needed for this WinIDE program, but might for others
- Start Simulator
 - click on Simulator (EXE2) icon 
 - should open several SIM12CPU Simulator windows
- Setup Simulator Windows
 - as needed, arrange & resize windows listed under Windows menu
- View memory values set by FCB directives
 - right-click inside Memory Window 1
 - select Set Based Address
 - enter 6000 and press OK (or hit enter); resize window if needed
- Question
 - Do you see the values set by the FCB statements in your code?
 - What does EQU do?
 - What does ORG do?
 - What does FCB do?

Example 2

- Open ex1.asm and Save File as ex2.asm
- Edit code (add/modify the purple below)

```
;331 Example 2 -Basic ASM
```

```
;**begin data block definition
```

```
TDR EQU $6000 ;top of data record
```

```
Top EQU $4000 ;top of program
```

```
ORG TDR ;set orgin
```

```
FCB $AA,$BB,$CC,$FF,$EE,$01
```

```
;**begin program
```

```
ORG Top
```

```
LDAA #$88
```

```
LDAB $6000
```


```
SWI
```

```
END
```

- Save file
- Questions
 - What does each of the new lines of code do?
 - What address mode(s) are used by the LDAA/B instructions?



Assembling & Simulating ASM Code

- Assemble ex2.asm
 - check/correct any syntax errors in .lst file
- Start Simulator
 - click on Simulator (EXE2) icon 
 - should open several SIM12CPU Simulator windows
- Set Program Counter (PC)
 - tell the simulator where your code begins by setting the PC
 - left-click on PC in CPU12 Window or type "PC 4000" in command line
- View source code disassembled
 - look at Code Window 1: Source
- View memory values set by directives
 - set Memory Window 1 to address 6000
- View register values
 - observe register values in CPU12 Window (resize if necessary)
- Questions
 - What are the initial values in accA? in accB?



Example 2 continued

- Simulate code
 - make sure PC is still set to 4000
 - check disassembled code in *Code Window 1*
 - click on *Go!* icon
- Program will run until it hits SWI or END
 - SWI may give error because no interrupt routine defined
 - just ignore this error
- View register values
 - observe new register values in *CPU12 Window*
- Questions
 - What value is in accA? in accB?
 - Is it what you expect?



Tracing Code -Using Step

- Close the simulator
- Restart simulator and reload ex2 code
- Set starting PC value
 - set PC to 4000
- Step through program
 - click on Step
 - Step will execute program on instruction at a time
- Confirm register values
 - Did you see register values change? Does it match program?
- Click Step again and confirm register values
- Questions
 - What is the PC value when accA becomes \$88?
 - How many bytes are in each instruction?



Tracing Code -Using Break Points

- Break points
 - commands to stop code a specific point in program memory
- Restart simulator and reload ex2 code
- Set PC to 4000; observe code in *Code Window 1*
- Set break point before LDAB instruction
 - *Where (what mem. addr.) does LDAB begin?*
 - left-click that line in *Code Window 1*
 - then right-click and select *Toggle Breakpoint...*
- Run simulation
- Question
 - *Did simulation stop at break point?*
 - *What are register values? Expected?*

*you must first left click,
then right click*



Example 3

- Get ex3.asm from class website; save to your ece331 directory
- Open ex3.asm in WinIDE
- Assemble
 - view .lst file
- Questions
 - Any errors?
 - Think about what this code should do.
- Simulate by tracing through one step at a time
 - observe register values at each instruction
 - What value is in accA and accB at the end of this program?
- Set a breakpoint at \$400D, reset PC to 4000 and Go!
 - What value is in accA at the break?

