EE 319K
Introduction to Embedded Systems

Lecture 1: Numbers, 9S12
Programming, Intro to TExaS

2-1

Announcements

- Cookie day next Tuesday!
  - Informal "connection" discussion over free cookies
- Office hours
  - M 3-4:30pm, W 1:30-3pm, and after class ?
    - Alternative: M 2-5pm and after class
  - Extra office hour: F 12:30-1:30 in JPs, starting Feb. 17
- Lab space
  - Main lab: ACA 1.106
    - TA office hours (as posted)
    - Checkouts, in your assigned lab time (signup sheet)
  - General lab usage
    - ACA 1.106 (M-F 9am-9pm)
    - ACA 1.108 (W,Th 2-6pm)
  - Limited space & hours, you need your own laptop!
- Survey
  - Incoming survey on Blackboard

Agenda

- ❑ Recap
  - ❖ Embedded systems
  - ❖ Product life cycle
  - ❖ Structured programming
- ❑ Outline
  - ❖ Data representations
    - o Numbers, decimal/hexadecimal, characters/ASCII
  - ❖ 9S12 architecture
  - ❖ 9S12 assembly
  - ❖ TExaS simulator

Numbers

- *Precision* is the number of distinct or different values.

Binary bits	Bytes	Alternatives
8	1	256
10	2	1024
12	2	4096
16	2	65536
20	3	1,048,576
24	3	16,777,216
30	4	1,073,741,824
32	4	4,294,967,296
n	[[n/8]]	2^n

Decimal Digits

- Decimal digits are used to specify precision of multimeters
  - 0,1,2,3,4,5,6,7,8,9 is a full decimal digit (10 choices)
  - 0,1 is a half decimal digit (2 choices)
  - 0,1,2,3 is three quarters decimal digit (4 choices)
  - + or - is a half decimal digit (2 choices)
  - + or - 0,1 is three quarters decimal digit (4 choices)
- Table gives THE DEFINITION of decimal digits. The specification of decimal digits goes 4, 4 1/2, 4 3/4, 5, with no other possibilities in between. The numbers 4.3 and 4 1/8 are not valid representations of decimal digits.

decimal digits	exact range	exact alternatives
3	0 to 999	1,000
3 1/2	0 to 1999	2,000
3 3/4	0 to 3999	4,000
4	0 to 9999	10,000
4 1/2	0 to 19,999	20,000
4 3/4	0 to 39,999	40,000
5	0 to 99,999	100,000
5 1/2	0 to 199,999	200,000
5 3/4	0 to 399,999	400,000
N	0 to 10^N-1	10^N
N 1/2	0 to $2*10^N$-1	$2*10^N$
N 3/4	0 to $4*10^N$-1	$4*10^N$

Decimal Digits

3 1/2 decimal digits
Range:
000.0k,000.1k,…199.9k
Resolution:0.1k
Precision: 2000 alternatives

3 3/4 decimal digits
Range:
000.0k,000.1k,…399.9k
Resolution:0.1k
Precision: 4000 alternatives

4 1/2 decimal digits
Range:
000.00k,00.01k,…199,99k
Resolution:0.01k
Precision: 20000 alternatives

Hexadecimal Representation

environment	binary	hex	decimal
Freescale	%01111010	$7A	122
Intel and TI	01111010B	7AH	122
C language		0x7A	122
LC-3		x7A	122

Convert from Binary to Hex

binary %11011001111101

nibbles 0011 0110 0111 1101

hexadecimal $367D

Convert from Hex to Binary

hexadecimal $1B3F

nibbles 0001 1011 0011 1111

binary %0001101100111111

8-bit Unsigned numbers

b7	b6	b5	b4	b3	b2	b1	b0

$N = 128b_7 + 64b_6 + 32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0$

The *basis* of a number system is a subset from which linear combinations of the basis elements can be used to construct the entire set: {1, 2, 4, 8, 16, 32, 64, 128}

Example: What is the unsigned binary for 175

Number	Basis	Need it?	bit	Operation
175	128	yes	bit 7=1	subtract 175-128
47	64	no	bit 6=0	none
47	32	yes	bit 5=1	subtract 47-32
15	16	no	bit 4=0	none
15	8	yes	bit 3=1	subtract 15-8
7	4	yes	bit 2=1	subtract 7-4
3	2	yes	bit 1=1	subtract 3-2
1	1	yes	bit 0=1	subtract 1-1

8-bit Signed numbers

b7	b6	b5	b4	b3	b2	b1	b0

$N = -128b_7 + 64b_6 + 32b_5 + 16b_4 + 8b_3 + 4b_2 + 2b_1 + b_0$

basis : {1, 2, 4, 8, 16, 32, 64, -128}

Example: What is the unsigned binary for -90

Number	Basis	Need it	bit	Operation
-90	-128	yes	bit 7=1	subtract -90 - -128
38	64	no	bit 6=0	none
38	32	yes	bit 5=1	subtract 38-32
6	16	no	bit 4=0	none
6	8	no	bit 3=0	none
4	4	yes	bit 2=1	subtract 6-4
2	2	yes	bit 1=1	subtract 2-2
0	1	no	bit 0=0	none

8-bit Signed numbers

Other methods:

- A second way to convert negative numbers into binary is to first convert them into unsigned binary, then do a two's complement negate (flip bits and add 1)
- A third way to convert negative numbers into binary is to first add 256 to the number, then convert the unsigned result to binary using the unsigned method.

What about 16-bit Unsigned/Signed Numbers:

For the unsigned 16-bit number system the basis is

{1, 2, 4, 8, 16, 32, 64, 128,
256, 512, 1024, 2048, 4096, 8192, 16384, 32768}

For the signed 16-bit number system the basis is

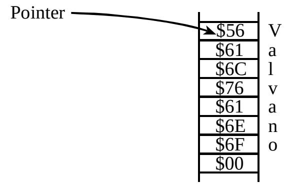
{1, 2, 4, 8, 16, 32, 64, 128,
256, 512, 1024, 2048, 4096, 8192, 16384, -32768}

Characters: ASCII

BITS 4 to 6

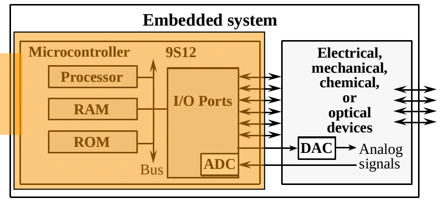
		0	1	2	3	4	5	6	7
	0	NUL	DLE	SP	0	@	P	`	p
B	1	SOH	DC1	!	1	A	Q	a	q
I	2	STX	DC2	"	2	B	R	b	r
T	3	ETX	DC3	#	3	C	S	c	s
S	4	EOT	DC4	$	4	D	T	d	t
	5	ENQ	NAK	%	5	E	U	e	u
0	6	ACK	SYN	&	6	F	V	f	v
	7	BEL	ETB	'	7	G	W	g	w
T	8	BS	CAN	(	8	H	X	h	x
O	9	HT	EM	)	9	I	Y	i	y
	A	LF	SUB	*	:	J	Z	j	z
3	B	VT	ESC	+	;	K	[	k	{
	C	FF	FS	,	<	L	\	l	;
	D	CR	GS	-	=	M	]	m	}
	E	SO	RS	.	>	N	^	n	~
	F	S1	US	/	?	O	_	o	DEL

Strings are stored as a Sequence of ASCII characters followed by a null



9S12 Assembly Programming

9S12DP512
9S12DG128



9S12DP512 Memory Map			
Address	**Size**	**Device**	**Contents**
$0000 to $03FF	1024	I/O Ports for Devices	Access external devices
$0400 to $07FF	1024	EEPROM: Elec. Erasable	Fixed Constants
($2000)$0800 to $3FFF	(8192) 14336	RAM: Random Access	Variables and Stack
$4000 to $FFFF	48152	EEPROM: Elec. Erasable	Programs and fixed constants

Ramesh Yerraballi

2-12

The LC3 computer from EE 306 had an address space of 64K and an addressability of 16-bit. The Memory Address Register (MAR) is 16 bits and the Memory Data Register (MDR) is also 16-bits.

Vs.

All 9S12 processors have 8-bit addressability. The MAR is still 16-bits but the MDR is 8-bits wide.

Big-endian and Little-endian

Big Endian

address	contents
$0050	$03
$0051	$E8

Little Endian

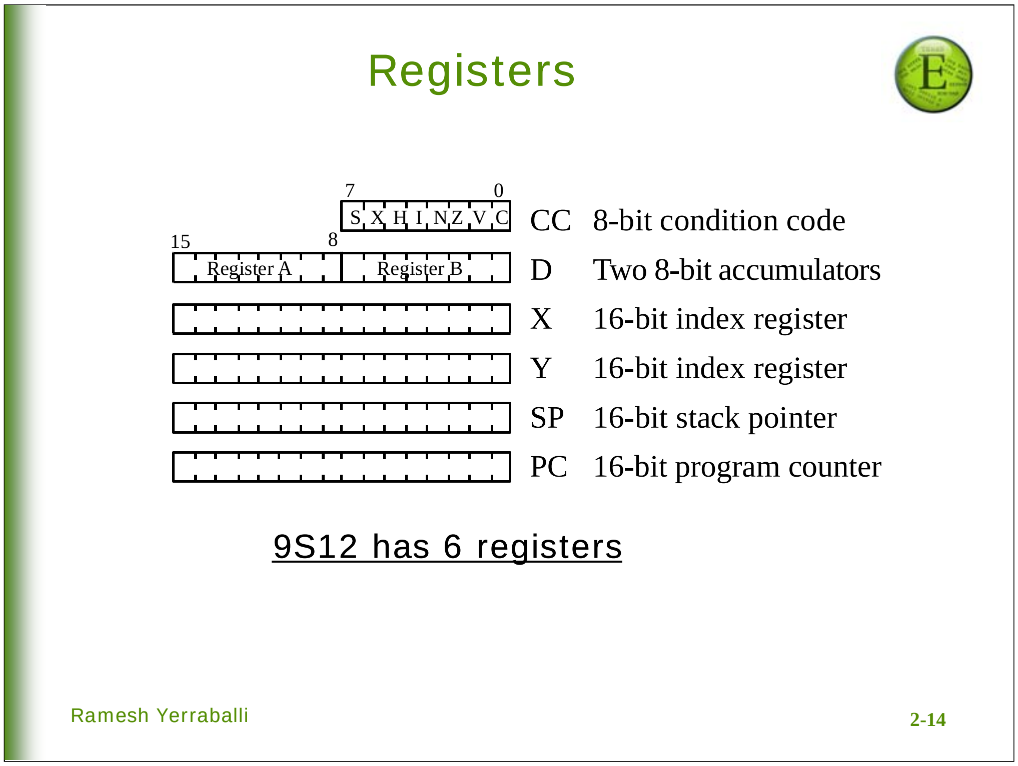
address	contents
$0050	$E8
$0051	$03

Big Endian

address	contents
$0050	$12
$0051	$34
$0052	$56
$0053	$78

Little Endian

address	contents
$0050	$78
$0051	$56
$0052	$34
$0053	$12



RegA = [$3800] means perform an 8-bit memory read from location $3800 and place the result into register A.

[$3800] = RegA means perform a 8-bit memory write placing the value in Register A into memory location $3800.

{$3800} = $1234 is equivalent to the two operations

[$3800] = $12

[$3801] = $34

Q: Does {$3800} = #1234 do the same?

A: No, that does [$3800] = $04; [$3801] = $D2; (1234_{10} = $04D2_{16}$)

The PC points to the address of the current instruction (Program Counter)

The CC (condition-code) register contains codes that reflect the results of the most recent instruction. Many “branch” type instructions “test” these bits for their operation.

The *stack* is s temporary storage implemented in the RAM. We push and pop elements onto and off the stack as we desire. The stack pointer (SP) keeps track of the current location of the “top” of the stack.

9S12 Programming

Simple Program

```
        org $3800
count  rmb 1
        org $4000
main   ldaa #10   ;loop counter
        staa count
loop   dec  count ; 9,8,… 0
        bne loop
        stop
        org $FFFE
        fdb main
```

Some Basic Instructions

```
ldaa #10   ;make A=10
ldaa $3800 ;A=contents of memory
staa $3800 ;Store contents of A
           ;into mem
deca       ;A=A-1
lsra       ;A=A/2 (shift right)
lsla       ;A=A*2 (shift left)
adda #10   ;A=A+10
anda #$02  ;A=A&2 (logic and)
eora #$08  ;A=A^8 (exclusive or)
oraa #$03  ;A=A|3 (logic or)
bra  loop  ;always jump to loop
bne  loop  ;jump to loop if
           ; not zero
beq  loop  ;jump to loop if zero

stop
```

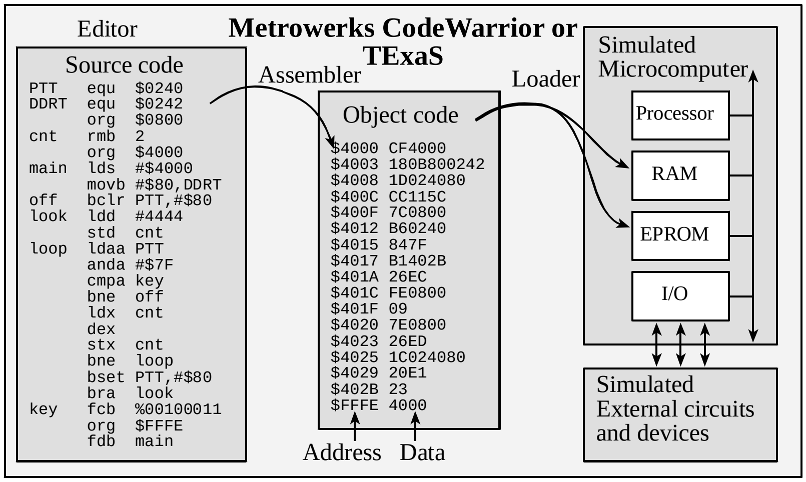
Ramesh Yerraballi

2-15

Pseudo ops: `org`, `rmb`, `equ`, `fdb` are not really part of the instruction set. They are macros that the assembler uses.

Labels like count main and loop are used to symbolically identify variables or memory addresses.

TExas Simulator



S/W Development using TExas

Six Phases:

- Defining the microcomputer type and memory configuration
- Editing the program source code
- Assembling source code and loading object code into memory
- Interfacing external components (I/O)
- Debugging the program by running it on the interactive simulator
- Debugging the program by running it on an actual board

Components:

- **Program** file (*.rtf) source code.
  - **TheLog.rtf** logs information, interactive debugger
  - **TheList.rtf** the assembly listing
  - **TheCRT.rtf** the input/output data of a CRT terminal
- **Microcomputer** file (*.uc): Internal Computer
- **I/O Device** file (*.io) external I/O devices
- Stack file (*.stk) holds temporary information
- Scope file (*.scp) used for debugging
- Plot file (*.plt) displays graphical information

Ramesh Yerraballi

2-17

Design Problem: NOT Gate:

```
PTT         equu        $0240
DDRT        equ         $0242
            org         $4000
main        ldaa        #$02 ; Set PTT0 as Switch and PTT1 as Light

            staa        DDRT
            ldaa        #0
            staa        PTT  ; Initially turn Light off
loop        ldaa        PTT
            anda        #$01 ; Mask all but the Switch inpu
            bne         press
            ldaa        #$02
            staa        PTT   ; turn Light ON if switch not pressed
            bra         loop
press       ldaa        #$00        ; turn Light OFF if Switch
pressed
            staa        PTT
            bra         loop
            stop

            org         $FFFE
            fdb         main
```