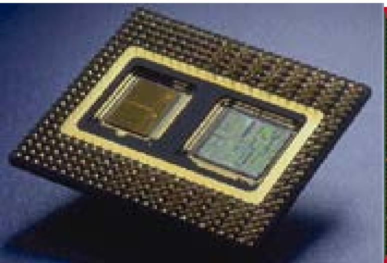
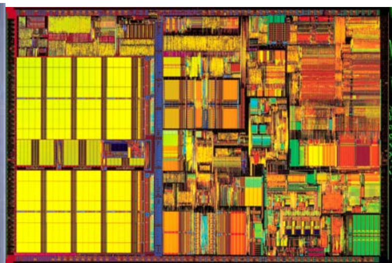
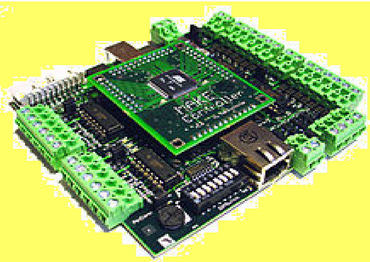
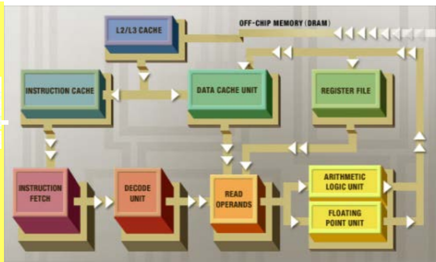
Lecture 7

Assembly Programming: Shift & Logical

Bryan Ackland
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030

Adapted from HCS12/9S12 An Introduction to Software and Hardware Interfacing   Han-Way Huang, 2010

Bit Condition (Masking) Branch Instructions

brclr (opr), (msk), (tar)

brset (opr), (msk), (tar)

where	opr	specifies the memory location to be checked
	msk	is an 8-bit mask that specifies which bits in the memory location are to be checked. Only check those that correspond to a '1' in the mask.
	tar	is branch target (label)

brclr instruction will branch if all selected bits are clear (='0')

brset instruction will branch if all selected bits are set (='1')

for example:

```
brclr   $400, $80, abc   ; branch to "abc" if MSbit of [$400] is '0'
brset   $640, $55, xyz   ; branch to "xyz" if all even bits of [$640] are '1'
```

Loop Example 3

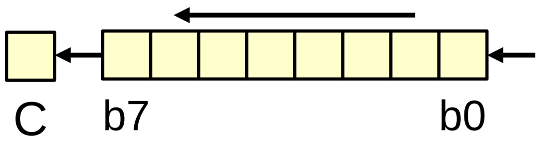
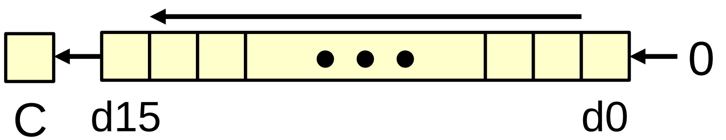
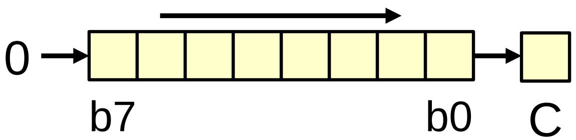
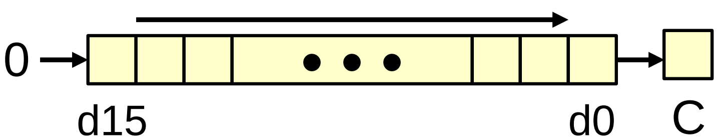
- Write a program to find the number of elements that are divisible by 4 in an array of N unsigned 8-bit elements starting at address $800.

```
N:      EQU     20
        ORG     $800
array:  DC.B    2,3,4,8,12,13,19,24,33,32
        DC.B    20,18,53,7,16,82,90,94,100,102
        ORG     $900
total:  DS.B    1

        ORG     $4000
        clr     total               ;init total to 0
        ldx     #array              ;set X point to array[0]
        ldab    #N                  ;set loop count to N
loop:   brclr   1, x+, $03, yes     ;check bits 1 and 0 & incr pointer
        bra     skip                ;not divisible by 4
yes:    inc     total               ;is divisible by 4
skip:   dbne    b, loop             ;done yet?
here:   bra     here                ;stay here
```

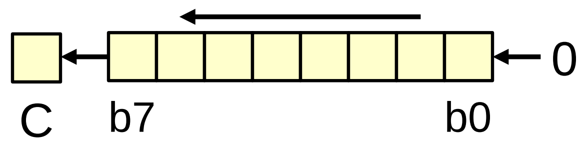
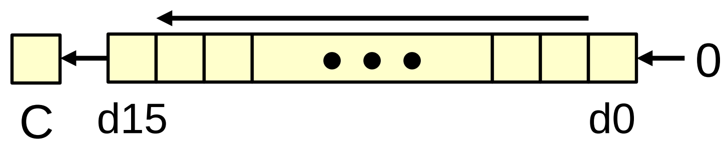
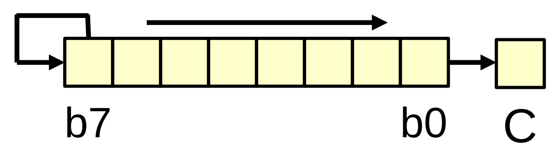
Shift and Rotate Instructions

- Shift and rotate instructions shift the operand right or left by 1 bit
- HCS12 does not support multi-bit shift instructions (barrel shifter)
- Carry bit “catches” bit shifted out to allow multi-precision shifts
- 3 types of shift: logical, arithmetic and rotate

Mnemonic	Function	Operation
lsl <opr> lsla lslb	Logical shift left memory Logical shift left A Logical shift left B	
lsld	Logical shift left D	
lsr <opr> lsra lsrb	Logical shift right memory Logical shift right A Logical shift right B	
lsrd	Logical shift right D	

Arithmetic Shift Instructions

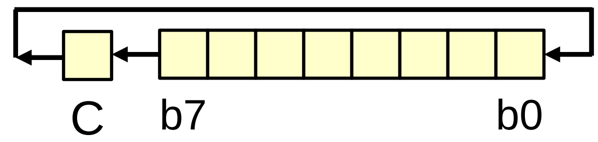
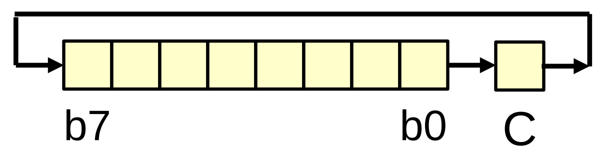
- arithmetic shift left is same as logical shift left
- arithmetic shift right preserves sign
- can be used to perform multiply and divide by 2 of signed data

Mnemonic	Function	Operation
asl <opr> asla aslb	Arithmetic shift left memory Arithmetic shift left A Arithmetic shift left B	
asld	Arithmetic shift left D	
asr <opr> asra asrb	Arithmetic shift right memory Arithmetic shift right A Arithmetic shift right B	

- There is no “arithmetic right shift D” instruction

Rotate Instructions

- rotate bits through the carry
- used to complete multi-precision shifts

Mnemonic	Function	Operation
rol <opr> rola rolb	Rotate left memory thru carry Rotate left A thru carry Rotate left B thru carry	
ror <opr> rora rorb	Rotate right memory thru carry Rotate right A thru carry Rotate right B thru carry	

- What two-instruction sequence could be used to perform a 16-bit arithmetic right shift on accumulator D?

Shift Example 1

- Write a program to count the number of 0's in the 16-bit number stored at $800~$801

```
        ORG     $800
numb:   DC.W    $2355
        ORG     $900
zeros:  DS.B    1
lp_cnt: DS.B    1

        ORG     $4000
        clr     zeros           ;init zero count  to 0
        movb    #16, lp_cnt     ;set up loop count
        ldd     numb            ;place data in D
loop:   lsrd                    ;shift lsb to C
        bcs     skip            ;branch if C is 1
        inc     zeros           ;inc zero count
skip:   dec     lp_cnt          ;increment pointer
        bne     loop            ;done yet?
here:   bra     here            ;wait here when done
```

Shift Example 2

- Write a program to logically shift the 32-bit number stored at $1000~$1003 to the right 5 places.

```
        ORG     $4000
        ldab    #5              ;set up loop count
        ldx     #$1000          ;data pointer
sloop:  lsr     0, x            ;shift MSbyte
        ror     1, x            ;shift next byte
        ror     2, x            ;shift next byte
        ror     3, x            ;shift LSbyte
        dbne    b, sloop        ;done yet?
        bgnd
```

Boolean Logic Instructions

- Allows for simple bit manipulations on 8-bit operands
- Important in I/O operations

Mnemonic	Function	Operation
anda <opr>	AND A with memory	A ← [A] • [M]
andb <opr>	AND B with memory	B ← [B] • [M]
andcc <opr>	AND CCR with memory	CCR ← [CCR] • [M]
eora <opr>	XOR A with memory	A ← [A] ⊕ [M]
eorb <opr>	XOR B with memory	B ← [B] ⊕ [M]
oraa <opr>	OR A with memory	A ← [A] + [M]
orab <opr>	OR B with memory	B ← [B] + [M]
oracc <opr>	OR CCR with memory	CCR ← [CCR] + [M]
clc	Clear C bit in CCR	C ← 0
cli	Clear I bit in CCR	I ← 0
clv	Clear V bit in CCR	V ← 0

Bitwise Logic Operations

- All Boolean logic instructions see each operand as a collection of eight unrelated bits
- for $0 \le i \le 7$, i^{th} bit of first operand is and'd (or'd, xor'd) with i^{th} bit of second operand to produce i^{th} bit of the result
- *For example:*

```
ldaa  #$53    ; A is          0 1 0 1 0 0 1 1
anda  #$E2    ; and'd with    1 1 1 0 0 0 1 0
staa  $1000   ; result is     0 1 0 0 0 0 1 0
```

- What would the following instructions do?

```
orab  #$FF

oraa  #0

anda  #$F0

xorb  #$FF
```

Bit Manipulate Instructions

- Allow us to set or clear specific bits in a memory location while leaving the other bits unchanged

Mnemonic	Function	Operation
bclr <opr>, mask	Clear bits in memory	$M \leftarrow [M] \bullet \overline{mask}$
bset <opr>, mask	Set bits in memory	$M \leftarrow [M] + mask$

- Can only be applied to an 8-bit memory location
- *mask* is an immediate 8-bit value that specifies which bits to clear or set (1=clear or set bit, 0=leave bit unchanged)
- For example:

```
bclr    abc, $3F     ; clear all but the two MSbits of location labeled abc
bset    0, Y, $05    ; set bits 0 and 3 of location pointed to by Y
```

- Try not to confuse with instructions brclr and brset