Microcomputers I – CE 320

Mohammad Ghamari, Ph.D.

Electrical and Computer Engineering

Kettering University

Lecture 12 : Boolean Logic Instructions

Announcement

- Lecture 11 is uploaded on the blackboard.

- You are going to have your midterm exam on Thursday, Nov 16.

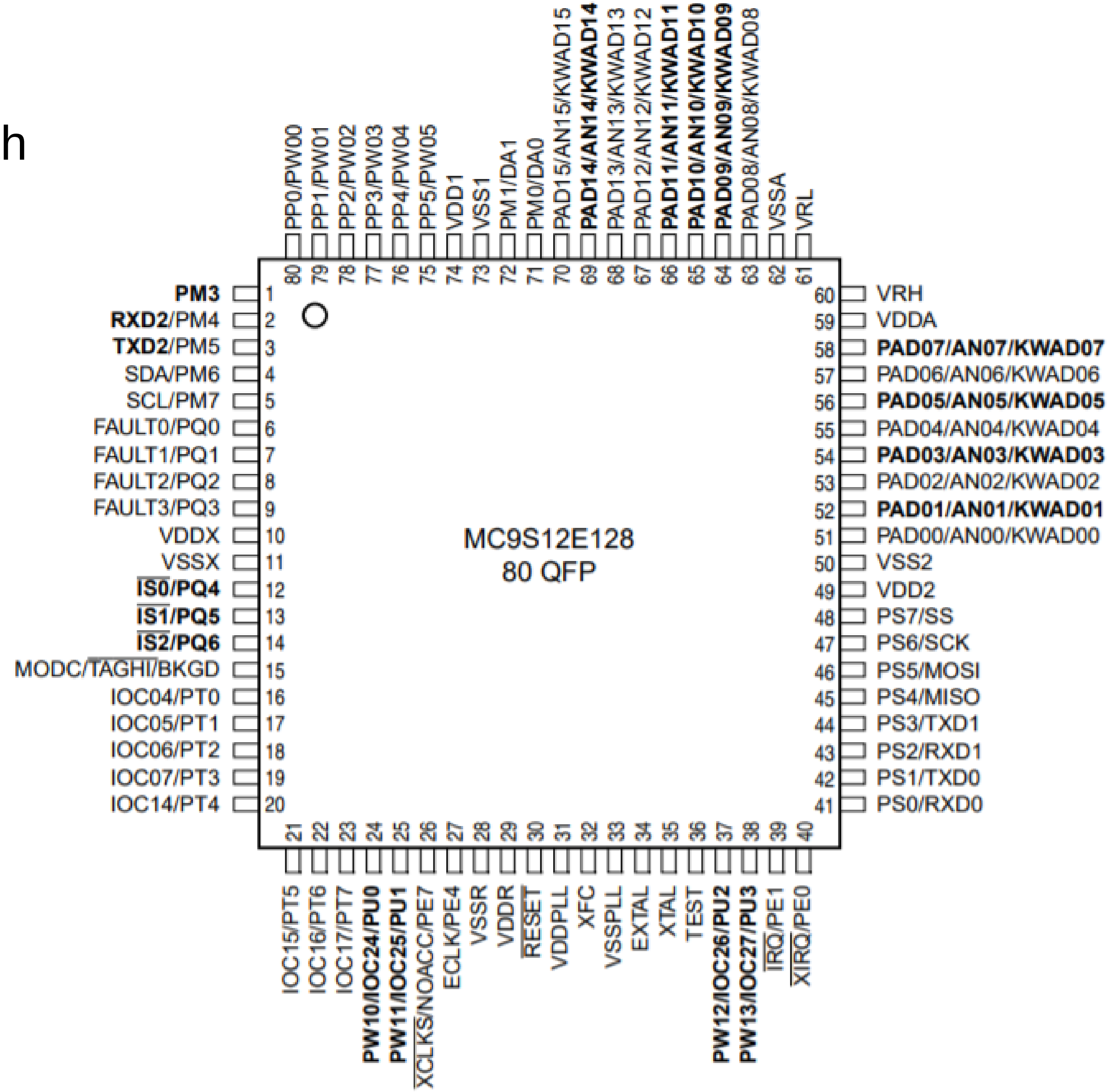
Today’s Topics

- Learn how to use Boolean instructions in assembly code

HCS12 Architecture Details.

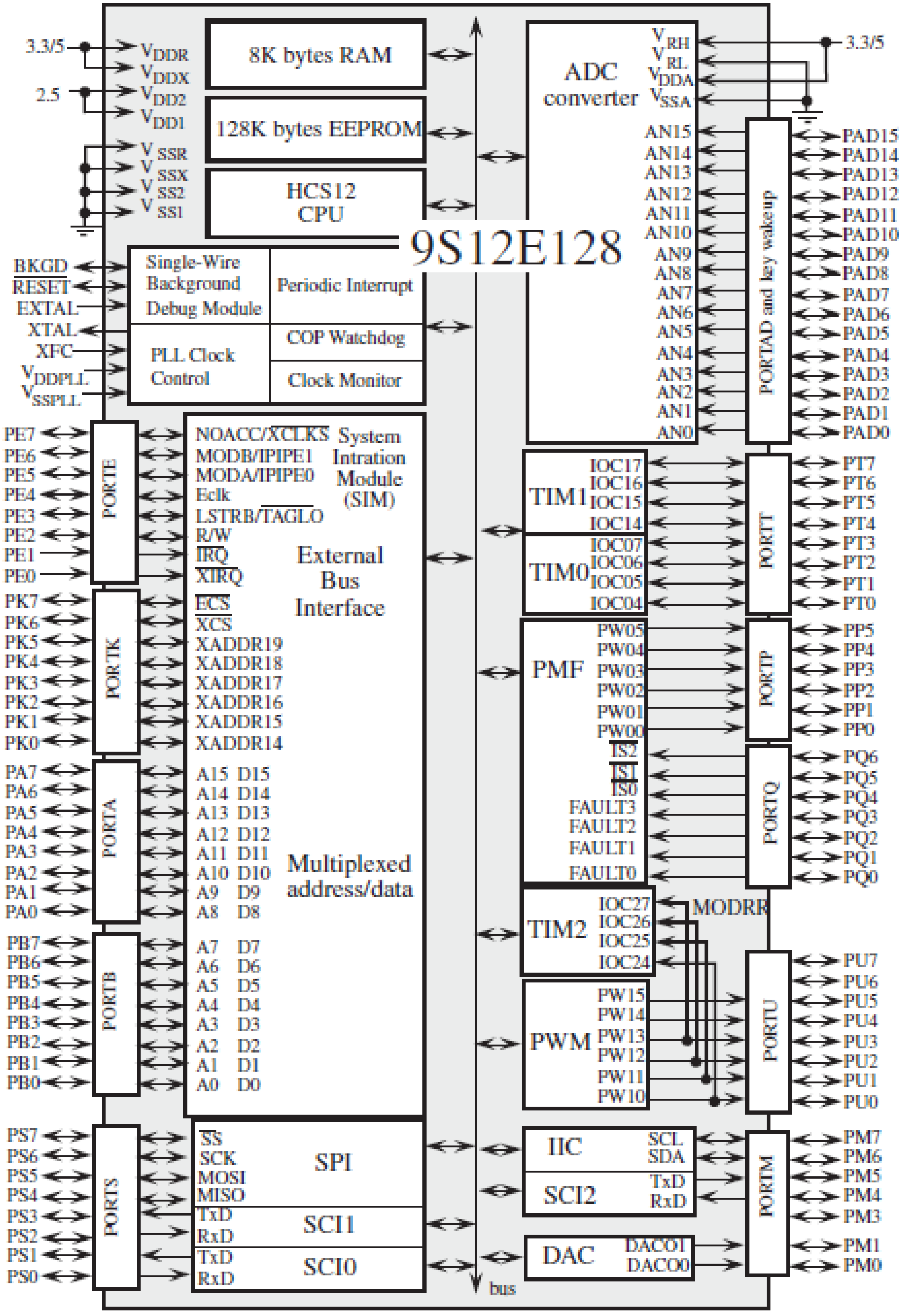
MC9S12E128 Pin Assignments

- There are two sizes of the 9S12E128 chip, one with **80 pins** (see fig) and the other with 112 pins
    - Ex: 112-pin chip has 92 **I/O pins**



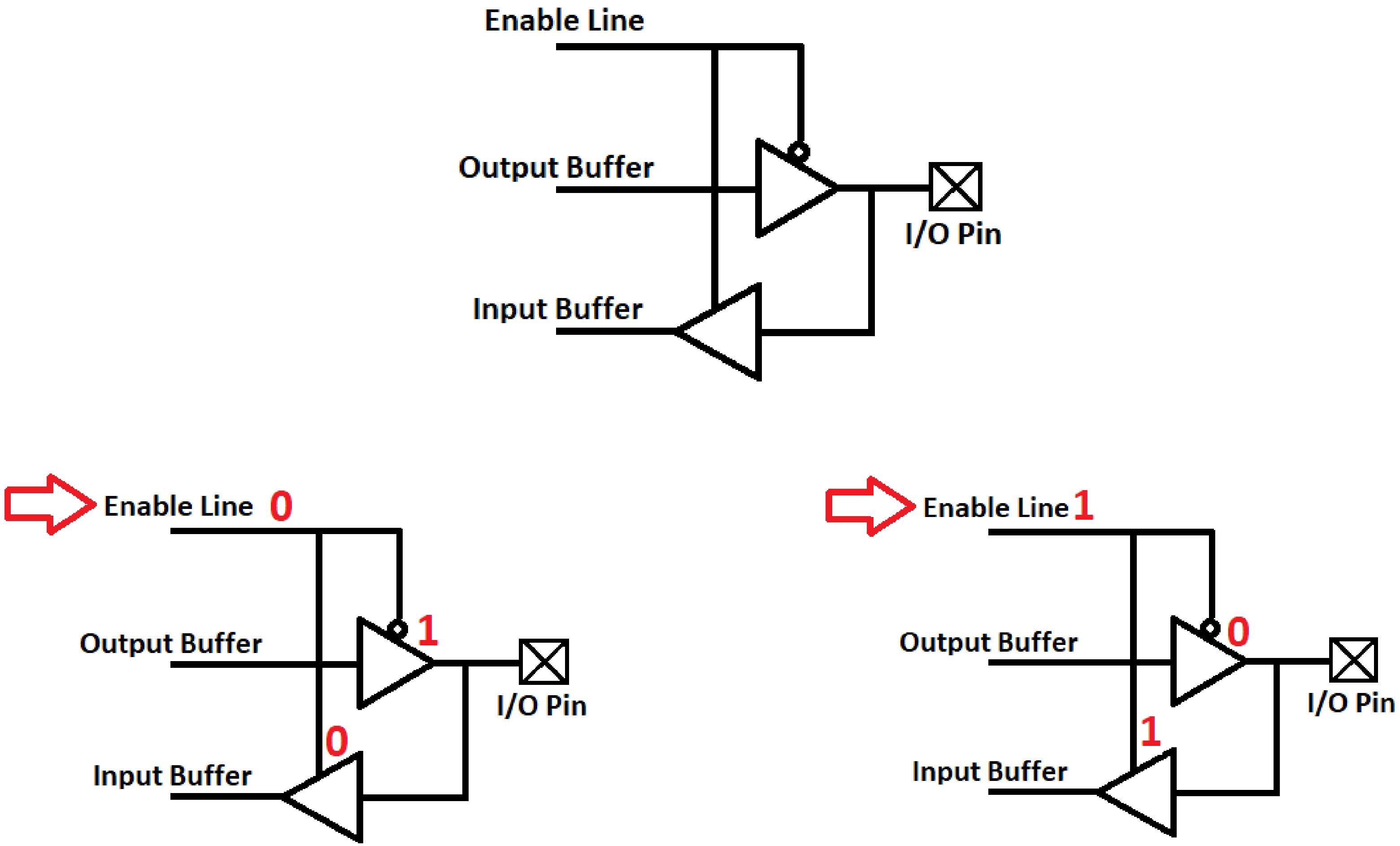
HCS12 Architecture Details. 9S12E128

- When dealing with **input and output port pins**, we often need to change the values of a few bits.
  - To make a specific pin in a port to behave as an **input**, we need to **clear (0)** that pin in the direction register.
    - Ex: push button, keypad, sensor signal
  - To make s specific pin in a port to behave as an output, we need to set (1) that pin in the direction register.
    - Ex: LED, DC motor
- For these types of applications, **Boolean logic instructions** come in handy.

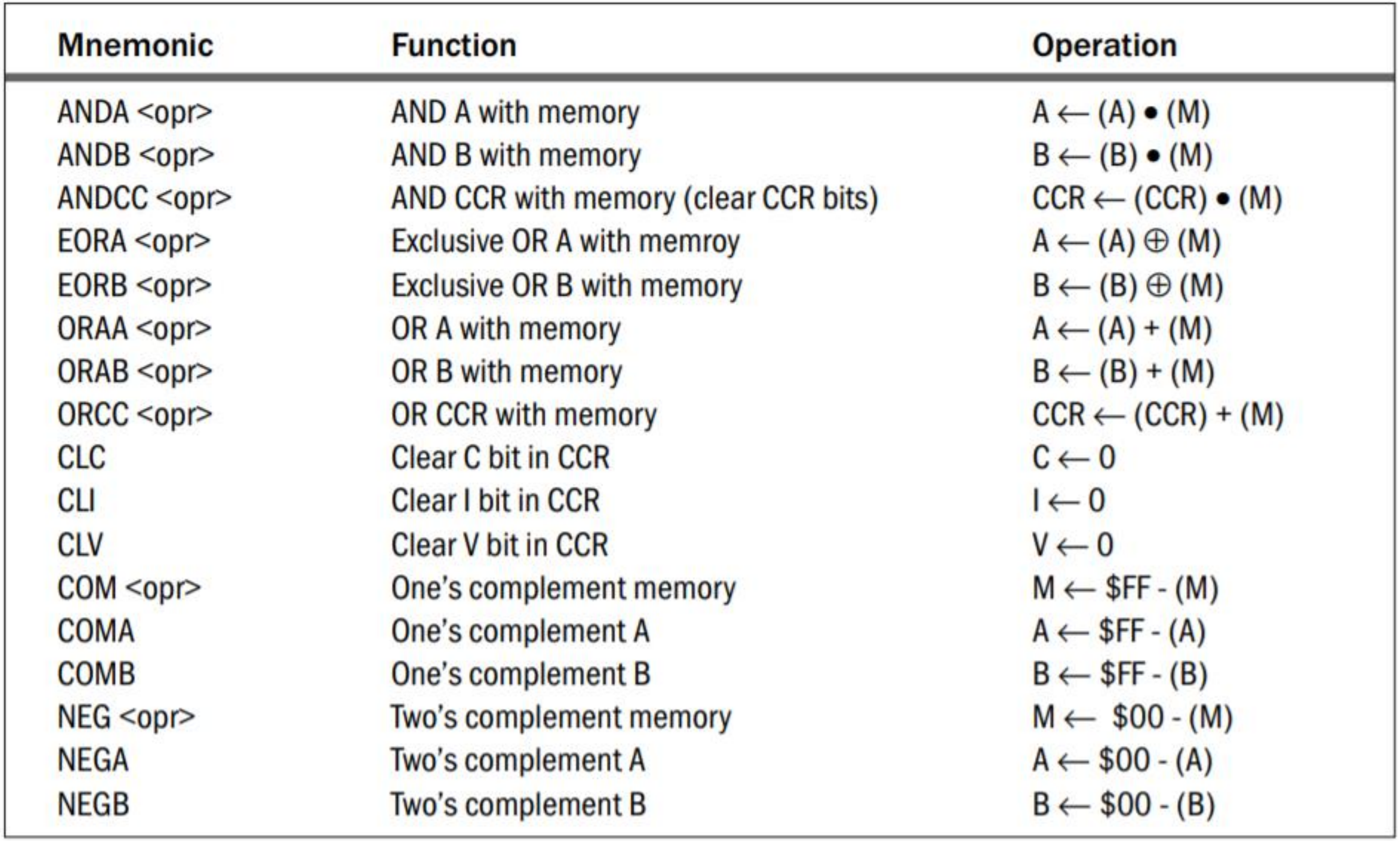


9S12E128
3.3/5
2.5
VDDR
VDDX
VDD2
VDD1
VSSR
VSSX
VSS2
VSS1
8K bytes RAM
128K bytes EEPROM
HCS12 CPU
ADC converter
VRH
VRL
VDDA
VSSA
3.3/5
AN15 AN14 AN13 AN12 AN11 AN10 AN9 AN8 AN7 AN6 AN5 AN4 AN3 AN2 AN1 AN0
PORTAD and key wakeup
PAD15 PAD14 PAD13 PAD12 PAD11 PAD10 PAD9 PAD8 PAD7 PAD6 PAD5 PAD4 PAD3 PAD2 PAD1 PAD0
BKGD
RESET
EXTAL
XTAL
XFC
VDDPLL
VSSPLL
Single-Wire Background Debug Module
Periodic Interrupt
COP Watchdog
PLL Clock Control
Clock Monitor
PE7 PE6 PE5 PE4 PE3 PE2 PE1 PE0
PORTE
NOACC/XCLKS
MODB/IPIPE1
MODA/IPIPE0
Eclk
LSTRB/TAGLO
R/W
IRQ
XIRQ
System Intration Module (SIM)
External Bus Interface
PK7 PK6 PK5 PK4 PK3 PK2 PK1 PK0
PORTK
ECS
XCS
XADDR19 XADDR18 XADDR17 XADDR16 XADDR15 XADDR14
PA7 PA6 PA5 PA4 PA3 PA2 PA1 PA0
PORTA
A15 D15 A14 D14 A13 D13 A12 D12 A11 D11 A10 D10 A9 D9 A8 D8
Multiplexed address/data
PB7 PB6 PB5 PB4 PB3 PB2 PB1 PB0
PORTB
A7 D7 A6 D6 A5 D5 A4 D4 A3 D3 A2 D2 A1 D1 A0 D0
PS7 PS6 PS5 PS4 PS3 PS2 PS1 PS0
PORTS
SS SCK MOSI MISO
SPI
TxD RxD
SCI1
TxD RxD
SCI0
TIM1
IOC17 IOC16 IOC15 IOC14
TIM0
IOC07 IOC06 IOC05 IOC04
PORTT
PT7 PT6 PT5 PT4 PT3 PT2 PT1 PT0
PMF
PW05 PW04 PW03 PW02 PW01 PW00
IS2 IS1 IS0
FAULT3 FAULT2 FAULT1 FAULT0
PORTP
PP5 PP4 PP3 PP2 PP1 PP0
PORTQ
PQ6 PQ5 PQ4 PQ3 PQ2 PQ1 PQ0
TIM2
IOC27 IOC26 IOC25 IOC24
MODRR
PWM
PW15 PW14 PW13 PW12 PW11 PW10
PORTU
PU7 PU6 PU5 PU4 PU3 PU2 PU1 PU0
IIC
SCL SDA
SCI2
TxD RxD
DAC
DACO1 DACO0
PORTM
PM7 PM6 PM5 PM4 PM3 PM1 PM0
bus

Behavior of GPIO Pins Inside MCU



Summary of Boolean logic instructions



Mnemonic	Function	Operation
ANDA <opr>	AND A with memory	$A \leftarrow (A) \bullet (M)$
ANDB <opr>	AND B with memory	$B \leftarrow (B) \bullet (M)$
ANDCC <opr>	AND CCR with memory (clear CCR bits)	$CCR \leftarrow (CCR) \bullet (M)$
EORA <opr>	Exclusive OR A with memroy	$A \leftarrow (A) \oplus (M)$
EORB <opr>	Exclusive OR B with memory	$B \leftarrow (B) \oplus (M)$
ORAA <opr>	OR A with memory	$A \leftarrow (A) + (M)$
ORAB <opr>	OR B with memory	$B \leftarrow (B) + (M)$
ORCC <opr>	OR CCR with memory	$CCR \leftarrow (CCR) + (M)$
CLC	Clear C bit in CCR	$C \leftarrow 0$
CLI	Clear I bit in CCR	$I \leftarrow 0$
CLV	Clear V bit in CCR	$V \leftarrow 0$
COM <opr>	One's complement memory	$M \leftarrow \$FF - (M)$
COMA	One's complement A	$A \leftarrow \$FF - (A)$
COMB	One's complement B	$B \leftarrow \$FF - (B)$
NEG <opr>	Two's complement memory	$M \leftarrow \$00 - (M)$
NEGA	Two's complement A	$A \leftarrow \$00 - (A)$
NEGB	Two's complement B	$B \leftarrow \$00 - (B)$

Logical Instructions

- One of the main purposes of the logical functions is to **affect individual bits** of a byte **without affecting** the others.

- These functions involve a **target** byte with the data and a **mask** byte which determines which bits are affected and which aren't.

- The table shows the affect of the values in the mask byte.
    - We will look at each of these functions along with examples

Function	0 Mask Bit	1 Mask Bit
AND	Clear to 0	No affect
OR	No affect	Set to 1
XOR	No affect	Toggle

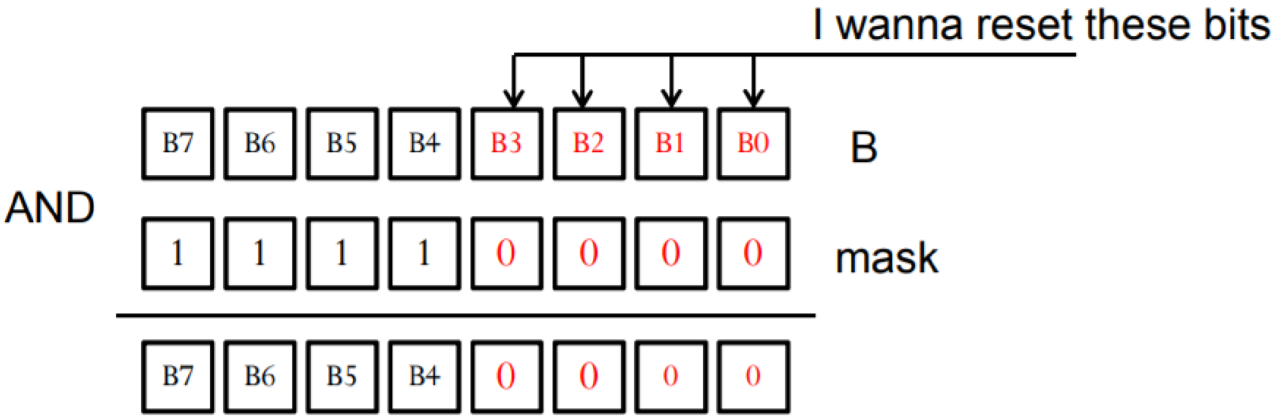
AND Operation

There are just two true AND functions.

- **ANDA, ANDB**
  - Sets N, Z, and clears V. No affect on C bit

- “AND” instruction is used to **clear** one or a few bits.

- Example: To clear the first 4 bits in register B,



I wanna reset these bits

	B7	B6	B5	B4	B3	B2	B1	B0	B
AND	1	1	1	1	0	0	0	0	mask
	B7	B6	B5	B4	0	0	0	0	

Thanks to:
Bi **AND** 0 = 0
Bi **AND** 1 = Bi

AND Operation

- Example: How to write instruction sequence to **clear** the upper four pins of the I/O port located at $56?

AND Operation

- Example: How to write instruction sequence to **clear** the upper four pins of the I/O port located at $56?

- **Solution:**

```
ldaa    $56
anda    #$0F
staa    $56
```

Alternatively, You can force bits 4,5,6,7 of M to be 0's

```
ldaa   M
anda   #%00001111
staa   M
```

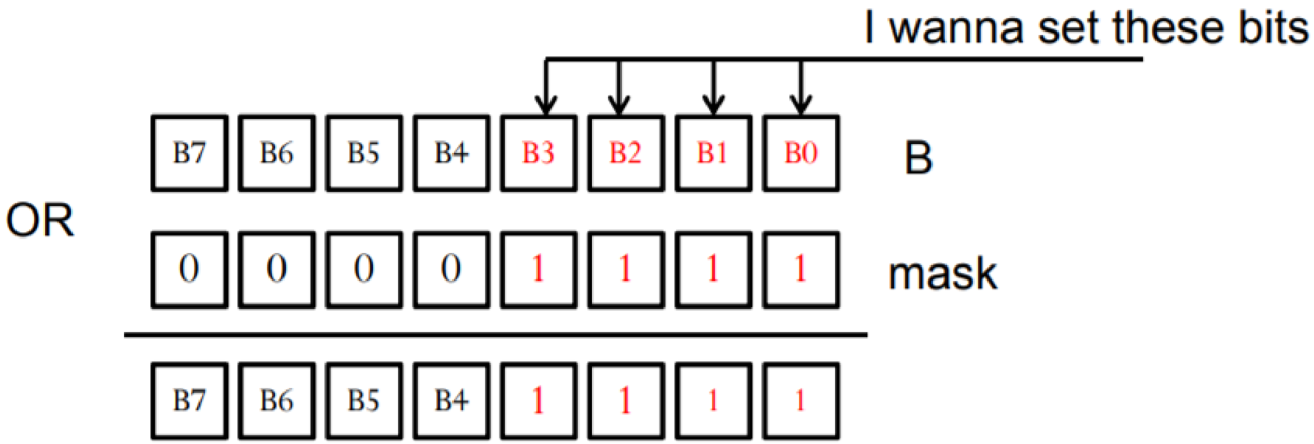
OR Operation

There are just two OR functions.

- **ORAA**, **ORAB**
  - Sets N, Z, and clears V. No affect on C bit

- “OR” instruction is used to **set** one or a few bits.

- Example: To set the first 4 bits in register B,



I wanna set these bits

	B7	B6	B5	B4	B3	B2	B1	B0	B
OR	0	0	0	0	1	1	1	1	mask
	B7	B6	B5	B4	1	1	1	1	

Thanks to:
Bi **OR** 0 = Bi
Bi **OR** 1 = 1

OR Operation

- Example: How to write instruction sequence to **set** the bit 0 of the I/O port located at $56?

OR Operation

- Example: How to write instruction sequence to **set** the bit 0 of the I/O port located at $56?

- Solution:

```
ldaa   $56
oraa   #$01
staa   $56
```

Alternatively, You can force bit 0 of M to be 1

```
ldaa     M
oraa     #%00000001
staa     M
```

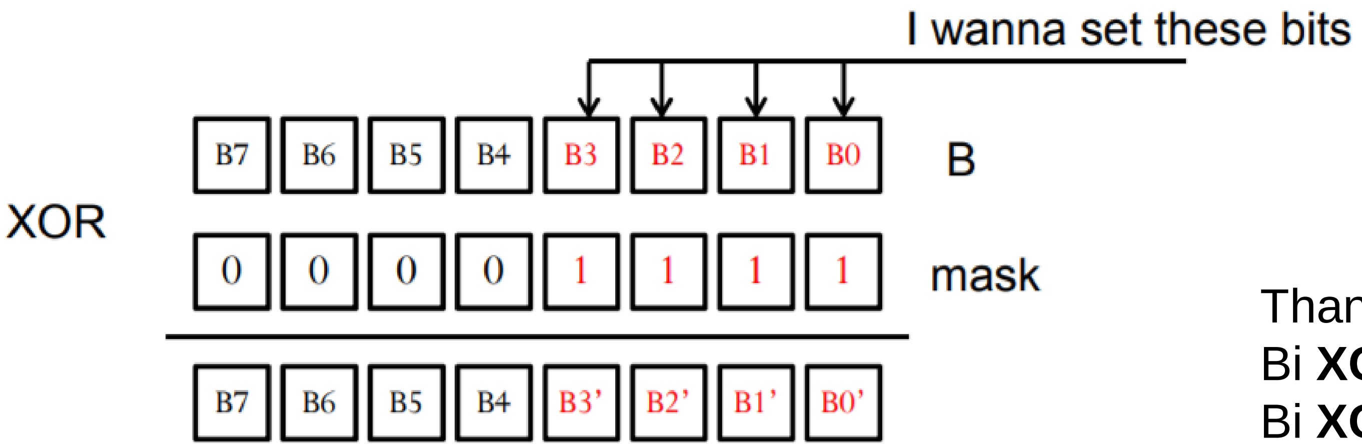
XOR Operation

There are just two exclusive-OR functions.

- **EORA, EORB**
  - Sets N, Z, and clears V. No affect on C bit

- “XOR” instruction is used to **flip** (change 0 to 1 and 1 to 0) one or more bits.

- Example: To flip the first 4 bits in register B,



I wanna set these bits

	B7	B6	B5	B4	B3	B2	B1	B0	B
XOR	0	0	0	0	1	1	1	1	mask
	B7	B6	B5	B4	B3’	B2’	B1’	B0’	

Thanks to:
Bi **XOR** 0 = Bi
Bi **XOR** 1 = Bi’
Bi’ = the inversion of Bi

XOR Operation

- Example: How to write instruction sequence to **toggles** the lower four bits of the I/O port at $56?

XOR Operation

- Example: How to write instruction sequence to **toggles** the lower four bits of the I/O port at $56?

- Solution:

```
ldaa        $56
eora        #$0F
staa        $56
```

NOT Operation

There are three complement functions.

- **COMA, COMB, COM**
  - Sets N, Z, and clears V, and sets C
  - Can be used if **all** of the port pins need to be **toggled**
  - The only logical instruction that **does not use a mask byte**

NOT Operation

Example:

- How to negate accumulator A or B?
  - **COMA**
  - **COMB**

- How to negate accumulator A or B **without** using the logical complement functions?
  - **EORA   #%11111111**
  - **EORB   #%11111111**

Example

- Consider a two-door sports car with a trunk and a glove box.
  - Assume that contact switches are used to
    - Monitor each door and send signals to the processor indicating
      - whether the door is open (TRUE) or closed (FALSE)
    - Four bits are required to monitor two side doors, a trunk, and a glove box.
    - The four bits will be **7**, **6**, **5**, and **4** of memory location **$0000**.
    - Microprocessor can read the contents of this location at any time to read the **status** of the doors.

  - The microprocessor also maintains a bit for the glove box light, cabin light and the trunk light.
    - Storing a 0 in the bit will cause the light to be OFF
    - Storing a 1 turns the light ON.
    - These three bits will be **2**, **1**, and **0** of the location **$0001** respectively

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	-	-	-
$0001	-	-	-	-	-	GBOXL	CBNL	TRNKL

Example 1

- **Q:** How to turn off the glove box light without affecting the other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	-	-	-
$0001	-	-	-	-	-	**GBOXL**	CBNL	TRNKL

Example 1

- **Q:** How to turn off the glove box light without affecting the other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	-	-	-
$0001	-	-	-	-	-	**GBOXL**	CBNL	TRNKL

- **Ans:**

Turn OFF → Use **AND** with a proper mask byte

```
ldaa    $01
anda    #%11111011
staa    $01
```

Example 2

- **Q:** How to turn on the trunk light without affecting the other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	-	-	-
$0001	-	-	-	-	-	GBOXL	CBNL	**TRNKL**

Example 2

- **Q:** How to turn on the trunk light without affecting the other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	-	-	-
$0001	-	-	-	-	-	GBOXL	CBNL	**TRNKL**

- **Ans:**

Turn ON → Use **OR** with a proper mask byte

```
ldd     $01
oraa    #%00000001
staa    $01
```

Example 3

- **Q:** How to toggle the cabin light without affecting the other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	-	-	-
$0001	-	-	-	-	-	GBOXL	**CBNL**	TRNKL

Example 3

- **Q:** How to toggle the cabin light without affecting the other bits?

	7	6	5	4	3	2	1	0
$0000	GBOXD	LEFTD	RGHTD	TRNKD	-	-	-	-
$0001	-	-	-	-	-	GBOXL	**CBNL**	TRNKL

- **Ans:**

Toggle → Use **XOR** with a proper mask byte

```
ldaa        $01
eora        #%00000010
staa        $01
```

Homework Example

- Toggle the cabin lights at exactly 1000 Hz

```
flip:          LDAA    $00      ; 3
               EORA    #CBNL    ; 2
               STAA    $00      ; 3
               LDX     #N       ; 2
loop:          DEX              ; N
               BNE     loop     ; 3(N-1)+1
               BRA     flip     ; 3
```

- 1KHz → 1000 times / sec
- Clock speed of Dragon12+:
  - 24 MHz (24,000,000 Hz) means 24 million clock cycles / sec
- When the sum of all cycles of the lines become 24,000, we can say the module runs 1,000 times per second.
- 3 + 2 + 3 + 2 + N + 3(N-1) + 1 +3 = 24,000
  - 11 + 4N = 24,000 then, 4N = 23989. Therefore, N = 5997.25
  - N should be an integer, so 4N + 11 + ? = 24,000
  - If 5 is used for ?, then N = 5996

Homework Example - continued

- Toggle the cabin lights at exactly 1,000 Hz

```
flip:           LDAA    $00             ; 3
                EORA    #CBNL           ; 2
                NOP                     ; 1
                NOP                     ; 1 (to add 5 extra clock cycles)
                BRA     0               ; 3 (use 3 clock cycles while do nothing)
                STAA    $00             ; 3
                LDX     #5996           ; 2
loop:           DEX                     ; 5996
                BNE     loop            ; 3(5996-1)+1
                BRA     flip            ; 3
```

A Short Story about K and M in bytes

- In general,
  - K means 1,000
  - M means 1,000,000
- When you count bytes,
  - K means 1,024
  - M means 1,024 x 1,024
- 1,024 comes from
  - 2^{10} = 1,024
  - Remember 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, …

Questions?

Wrap-up

What we’ve learned

- Boolean logical instructions
- ANDx, ORAx, EORx, and COMx

What to Come

- Bit instructions

- Stack

- Subroutines