

ELEC 2220 Computer Systems

Chapter 3. Instruction Set

Soo-Young Lee
Department of Electrical and Computer Engineering
Auburn University

ELEC2220 Auburn University

3. Instruction Set: Chapter Objectives

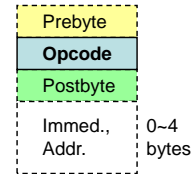
- How a computer operation is specified → Instruction
- Instruction Format
- Data Transfer Instructions
- Arithmetic Instructions
- Logic and Shift/Rotate Instructions
- Branch Instructions

Condition code bits ↔ Flags

ELEC2220 Auburn University

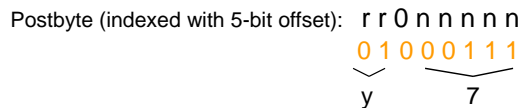
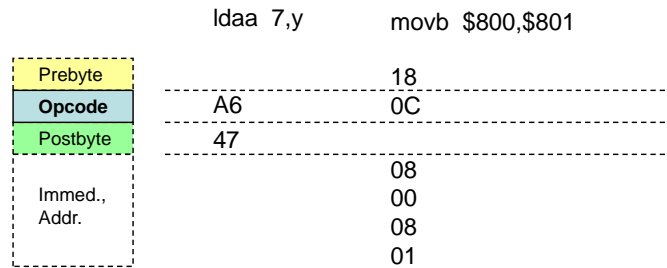
3-1. M68HC12 Architecture

- All memory and I/O are mapped in a common 64 Kbytes address space.
 - The same set of instructions can be used to access memory, I/O ports, and control registers.
- 8-bit and 16-bit operands
- Most instructions involve a register (including accumulator).
 - Load, Store, ...
- An instruction is 1 ~ 6 bytes long.
- An instruction may involve 0~2 operands.
- Instruction Format
 - Prebyte
 - Opcode
 - Postbyte
 - Immediate operand, addresses (offset, effective address)



3-1. M68HC12 Architecture

- Examples (Instruction Format)



3-2. Data Transfer Instructions

3-2-1. Load (LD r)

- Copy memory content into a register r .
 - Source: any addressing mode except for *inherent*
 - Destination: register (A,B,D,SP,X,Y)
- N, Z and V are updated (0 \rightarrow V).

- Load Effective Address (LEA r)
 - Load X, Y or SP with an effective address.
 - Source: indexed addressing only
 - No flag updated
 - Used to change EA in X, Y or SP

ELEC2220 Auburn University

3-2. Data Transfer Instructions

- Load (LD r), Load Effective Address (LEA r)

Load Instructions		
Mnemonic	Function	Operation
LDAA	Load A	(M) \Rightarrow A
LDAB	Load B	(M) \Rightarrow B
LDD	Load D	(M : M + 1) \Rightarrow (A:B)
LDS	Load SP	(M : M + 1) \Rightarrow SP
LDX	Load Index Register X	(M : M + 1) \Rightarrow X
LDY	Load Index Register Y	(M : M + 1) \Rightarrow Y
LEAS	Load Effective Address Into SP	Effective Address \Rightarrow SP
LEAX	Load Effective Address Into X	Effective Address \Rightarrow X
LEAY	Load Effective Address Into Y	Effective Address \Rightarrow Y

ELEC2220 Auburn University

3-2. Data Transfer Instructions

3-2-2. Store (STr)

- Copy the content of a register r to memory.
 - Source: register (A,B,D,SP, X, Y)
 - Destination: any addressing mode except for *immediate* and *inherent*
- N, Z and V are updated ($0 \rightarrow V$)

Store Instructions		
Mnemonic	Function	Operation
STAA	Store A	$(A) \Rightarrow M$
STAB	Store B	$(B) \Rightarrow M$
STD	Store D	$(A) \Rightarrow M, (B) \Rightarrow M + 1$
STS	Store SP	$(SP) \Rightarrow M : M + 1$
STX	Store X	$(X) \Rightarrow M : M + 1$
STY	Store Y	$(Y) \Rightarrow M : M + 1$

ELEC2220 Auburn University

3-2. Data Transfer Instructions

3-2-3. Transfer (TFR r,R) and Exchange (EXG r,R)

- TFR: transfer the content of a register r to another register R
- EXG: swap the contents of two registers, r & R
- A register can be CCR.
- Flags are not updated except when CCR is involved.

- You may use the following formats:
 - TAB, TBA: N, Z and V are updated ($0 \rightarrow V$)
 - TSX, TSY, TXS, TYS, XGDX, XGDY (M68HC11), no flag updated

- Sign Extension (SEX r,R)
 - Do sign-extension on r into R
 - r : A, B, CCR
 - R : D, X, Y, SP

ELEC2220 Auburn University

3-2. Data Transfer Instructions

- Transfer (TFR r,R), Exchange (EXG r,R), Sign Extension (SEX r,R)

Transfer Instructions		
Mnemonic	Function	Operation
TAB	Transfer A To B	$(A) \Rightarrow B$
TAP	Transfer A To CCR	$(A) \Rightarrow \text{CCR}$
TBA	Transfer B To A	$(B) \Rightarrow A$
TFR	Transfer Register To Register	$(A, B, \text{CCR}, D, X, Y, \text{ or } \text{SP}) \Rightarrow A, B, \text{CCR}, D, X, Y, \text{ or } \text{SP}$
TPA	Transfer CCR To A	$(\text{CCR}) \Rightarrow A$
TSX	Transfer SP To X	$(\text{SP}) \Rightarrow X$
TSY	Transfer SP To Y	$(\text{SP}) \Rightarrow Y$
TXS	Transfer X To SP	$(X) \Rightarrow \text{SP}$
TYS	Transfer Y To SP	$(Y) \Rightarrow \text{SP}$
Exchange Instructions		
Mnemonic	Function	Operation
EXG	Exchange Register To Register	$(A, B, \text{CCR}, D, X, Y, \text{ or } \text{SP}) \Leftrightarrow (A, B, \text{CCR}, D, X, Y, \text{ or } \text{SP})$
XGDX	Exchange D With X	$(D) \Leftrightarrow (X)$
XGDY	Exchange D With Y	$(D) \Leftrightarrow (Y)$
Sign Extension Instruction		
Mnemonic	Function	Operation
SEX	Sign Extend 8-bit Operand	$(A, B, \text{CCR}) \Rightarrow X, Y, \text{ or } \text{SP}$

ELEC2220 Auburn University

3-2. Data Transfer Instructions

3-2-4. Move (MOVB, MOVW src, dst)

- Move a byte or word from a memory location (including an immediate operand) to another location.
- No register is involved.
- Addressing modes: *Immediate, Extended, Indexed*
- No flag updated

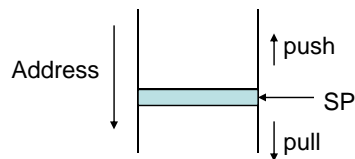
Mnemonic	Function	Operation
MOVB	Move Byte (8-bit)	$(M_1) \Rightarrow M_2$
MOVW	Move Word (16-bit)	$(M : M + 1_1) \Rightarrow M : M + 1_2$

ELEC2220 Auburn University

3-2. Data Transfer Instructions

3-2-5. Stack Instructions

- Stack
 - LIFO (Last-In-First-Out) data structure
 - The “top” byte is always pointed to by SP.



- The stack “grows” as operands (bytes, words) are pushed onto it and “shrinks” as they are pulled out of it.

ELEC2220 Auburn University

3-2. Data Transfer Instructions

- Push (PSH*r*)
 - Push (store) the content of a register *r* onto the stack.
 - $SP \leftarrow SP - \text{size of } r$
 - *r* is stored at the location pointed to by SP.
 - *r*: A, B, D, X, Y, CCR
 - No flags updated
- Pull (PUL*r*)
 - Pull (load) the byte or word at the top of stack into a register *r*.
 - *r* is loaded with the byte or word pointed by SP.
 - $SP \leftarrow SP + \text{size of } r$
 - *r*: A, B, D, X, Y, CCR
 - No flags updated (except when *r* is CCR)

ELEC2220 Auburn University

3-2. Data Transfer Instructions

- Push (PSH r), Pull (PUL r)

Stack Operation Instructions		
Mnemonic	Function	Operation
PSHA	Push A	$(SP) - 1 \Rightarrow SP; (A) \Rightarrow M_{(SP)}$
PSHB	Push B	$(SP) - 1 \Rightarrow SP; (B) \Rightarrow M_{(SP)}$
PSHC	Push CCR	$(SP) - 1 \Rightarrow SP; (A) \Rightarrow M_{(SP)}$
PSHD	Push D	$(SP) - 2 \Rightarrow SP; (A : B) \Rightarrow M_{(SP)} : M_{(SP+1)}$
PSHX	Push X	$(SP) - 2 \Rightarrow SP; (X) \Rightarrow M_{(SP)} : M_{(SP+1)}$
PSHY	Push Y	$(SP) - 2 \Rightarrow SP; (Y) \Rightarrow M_{(SP)} : M_{(SP+1)}$
PULA	Pull A	$(M_{(SP)}) \Rightarrow A; (SP) + 1 \Rightarrow SP$
PULB	Pull B	$(M_{(SP)}) \Rightarrow B; (SP) + 1 \Rightarrow SP$
PULC	Pull CCR	$(M_{(SP)}) \Rightarrow CCR; (SP) + 1 \Rightarrow SP$
PULD	Pull D	$(M_{(SP)} : M_{(SP+1)}) \Rightarrow A : B; (SP) + 2 \Rightarrow SP$
PULX	Pull X	$(M_{(SP)} : M_{(SP+1)}) \Rightarrow X; (SP) + 2 \Rightarrow SP$
PULY	Pull Y	$(M_{(SP)} : M_{(SP+1)}) \Rightarrow Y; (SP) + 2 \Rightarrow SP$

ELEC2220 Auburn University

3-3. Arithmetic Instructions

3-3-1. Addition (ArR , $ADCr$, $ADDr$) and Subtraction (SrR , $SBCr$, $SUBr$)

- 8-bit or 16-bit addition/subtraction is performed between two registers r & R , or between a register r and a memory operand.
- N, Z, V, C, & H are updated.
- $ADCr$ and $SBCr$ for multi-byte operands
 - If two-byte operands, $ADDD$ and $SUBD$
- DAA (Decimal Adjustment Addition) for BCD arithmetic (addition)
 - Packed BCD

3-3-2. Increment (INC , $INCr$, INr) and Decrement (DEC , $DECr$, DEr)

- A value of 1 is added to or subtracted from a register r or a memory operand.
- Flags are updated
 - N, Z, & V: DEC , $DECA$, $DECB$, INC , $INCA$, $INCB$
 - Z: DEX , DEY , INX , INY
 - DES & INS do not update any flag.

ELEC2220 Auburn University

3-3. Arithmetic Instructions

- Addition and Subtraction

Addition Instructions		
Mnemonic	Function	Operation
ABA	Add B To A	$(A) + (B) \Rightarrow A$
ABX	Add B To X	$(B) + (X) \Rightarrow X$
ABY	Add B To Y	$(B) + (Y) \Rightarrow Y$
ADCA	Add With Carry To A	$(A) + (M) + C \Rightarrow A$
ADCB	Add With Carry To B	$(B) + (M) + C \Rightarrow B$
ADDA	Add Without Carry To A	$(A) + (M) \Rightarrow A$
ADDB	Add Without Carry To B	$(B) + (M) \Rightarrow B$
ADDD	Add To D	$(A:B) + (M : M + 1) \Rightarrow A : B$
Subtraction Instructions		
Mnemonic	Function	Operation
SBA	Subtract B From A	$(A) - (B) \Rightarrow A$
SBCA	Subtract With Borrow From A	$(A) - (M) - C \Rightarrow A$
SBCB	Subtract With Borrow From B	$(B) - (M) - C \Rightarrow B$
SUBA	Subtract Memory From A	$(A) - (M) \Rightarrow A$
SUBB	Subtract Memory From B	$(B) - (M) \Rightarrow B$
SUBD	Subtract Memory From D (A:B)	$(D) - (M : M + 1) \Rightarrow D$

ELEC2220 Auburn University

3-3. Arithmetic Instructions

- Decrement and Increment

Decrement Instructions		
Mnemonic	Function	Operation
DEC	Decrement Memory	$(M) - \$01 \Rightarrow M$
DECA	Decrement A	$(A) - \$01 \Rightarrow A$
DECB	Decrement B	$(B) - \$01 \Rightarrow B$
DES	Decrement SP	$(SP) - \$0001 \Rightarrow SP$
DEX	Decrement X	$(X) - \$0001 \Rightarrow X$
DEY	Decrement Y	$(Y) - \$0001 \Rightarrow Y$
Increment Instructions		
Mnemonic	Function	Operation
INC	Increment Memory	$(M) + \$01 \Rightarrow M$
INCA	Increment A	$(A) + \$01 \Rightarrow A$
INCB	Increment B	$(B) + \$01 \Rightarrow B$
INS	Increment SP	$(SP) + \$0001 \Rightarrow SP$
INX	Increment X	$(X) + \$0001 \Rightarrow X$
INY	Increment Y	$(Y) + \$0001 \Rightarrow Y$

ELEC2220 Auburn University

3-3. Arithmetic Instructions

3-3-3. Compare and Test

- Perform subtraction to update flags without storing the result.
- Compare (CMP r , CP r)
 - Compare a register r (A,B,D,SP,X,Y) to another register (B) or a memory operand.
- Test (TST, TST r)
 - Compare a register r (A,B) or a memory operand to zero.
- Flags (Condition code bits)
 - Compare: N, Z, V, & C
 - Test: N, Z (0 \rightarrow V, C)

ELEC2220 Auburn University

3-3. Arithmetic Instructions

• Compare and Test

Compare Instructions		
Mnemonic	Function	Operation
CBA	Compare A To B	(A) - (B)
CMPA	Compare A To Memory	(A) - (M)
CMPB	Compare B To Memory	(B) - (M)
CPD	Compare D To Memory (16-bit)	(A : B) - (M : M + 1)
CPS	Compare SP To Memory (16-bit)	(SP) - (M : M + 1)
CPX	Compare X To Memory (16-bit)	(X) - (M : M + 1)
CPY	Compare Y To Memory (16-bit)	(Y) - (M : M + 1)
Test Instructions		
Mnemonic	Function	Operation
TST	Test Memory For Zero Or Minus	(M) - \$00
TSTA	Test A For Zero Or Minus	(A) - \$00
TSTB	Test B For Zero Or Minus	(B) - \$00

ELEC2220 Auburn University

3-3. Arithmetic Instructions

3-3-4 Multiplication

- 8-bit multiplication: MUL (unsigned)
 - $A \times B \rightarrow D$
- 16-bit multiplication: EMUL (unsigned), EMULS (signed)
 - $D \times Y \rightarrow Y:D$
- Flags (Condition code bits)
 - N (MSB of result), Z, & C (bit 7 or 15) are updated.

3-3-5. Division

- 16-bit by 16-bit
 - $D / X \rightarrow X$ (quotient), D (remainder)
 - IDIV (unsigned): Z, 0 \rightarrow V (always), 1 \rightarrow C if X=0 (divided by zero)
 - IDIVS (signed): N, Z, V, 1 \rightarrow C if X=0
- 32-bit by 16-bit
 - $Y:D / X \rightarrow Y$ (quotient), D (remainder)
 - EDIV (unsigned): N, Z, V, 1 \rightarrow C if X=0
 - EDIVS (signed): N, Z, V, 1 \rightarrow C if X=0

ELEC2220 Auburn University

3-3. Arithmetic Instructions

• Multiplication and Division

Multiplication Instructions		
Mnemonic	Function	Operation
EMUL	16 By 16 Multiply (Unsigned)	$(D) \times (Y) \Rightarrow Y : D$
EMULS	16 By 16 Multiply (Signed)	$(D) \times (Y) \Rightarrow Y : D$
MUL	8 By 8 Multiply (Unsigned)	$(A) \times (B) \Rightarrow A : B$
Division Instructions		
Mnemonic	Function	Operation
EDIV	32 By 16 Divide (Unsigned)	$(Y : D) \div (X)$ Quotient \Rightarrow Y Remainder \Rightarrow D
EDIVS	32 By 16 Divide (Signed)	$(Y : D) \div (X)$ Quotient \Rightarrow Y Remainder \Rightarrow D
FDIV	16 By 16 Fractional Divide	$(D) \div (X) \Rightarrow X$ remainder \Rightarrow D
IDIV	16 By 16 Integer Divide (Unsigned)	$(D) \div (X) \Rightarrow X$ remainder \Rightarrow D
IDIVS	16 By 16 Integer Divide (Signed)	$(D) \div (X) \Rightarrow X$ remainder \Rightarrow D

ELEC2220 Auburn University

3-3. Arithmetic Instructions

3-3-6. Clear, Complement and Negate

- Clear (CLC, CLI, CLV)
 - Set a bit in CCR, a memory operand, or a register r (A,B) to zero.
 - CLC, CLI, CLV
 - CLR m , CLRA, CLRB: N, V, & C are cleared, and Z is set.
- Complement (COM, COM r)
 - Derive the 1's complement of a memory operand or a register r (A,B).
 - N, Z, 0 \rightarrow V, 1 \rightarrow C
- Negate (NEG, NEG r)
 - Derive the 2's complement of a memory operand or a register r (A,B).
 - N, Z, V, C

ELEC2220 Auburn University

3-3. Arithmetic Instructions

• Clear, Complement and Negate

Mnemonic	Function	Operation
CLC	Clear C Bit In CCR	$0 \Rightarrow C$
CLI	Clear I Bit In CCR	$0 \Rightarrow I$
CLR	Clear Memory	$\$00 \Rightarrow M$
CLRA	Clear A	$\$00 \Rightarrow A$
CLRB	Clear B	$\$00 \Rightarrow B$
CLV	Clear V bit in CCR	$0 \Rightarrow V$
COM	One's Complement Memory	$\$FF - (M) \Rightarrow M$ or $(\bar{M}) \Rightarrow M$
COMA	One's Complement A	$\$FF - (A) \Rightarrow A$ or $(\bar{A}) \Rightarrow A$
COMB	One's Complement B	$\$FF - (B) \Rightarrow B$ or $(\bar{B}) \Rightarrow B$
NEG	Two's Complement Memory	$\$00 - (M) \Rightarrow M$ or $(\bar{M}) + 1 \Rightarrow M$
NEGA	Two's Complement A	$\$00 - (A) \Rightarrow A$ or $(\bar{A}) + 1 \Rightarrow A$
NEGB	Two's Complement B	$\$00 - (B) \Rightarrow B$ or $(\bar{B}) + 1 \Rightarrow B$

ELEC2220 Auburn University

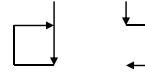
3-4. Branch Instructions

Program execution is not always *sequential*. → branching

3-4-1. Unconditional and conditional branches

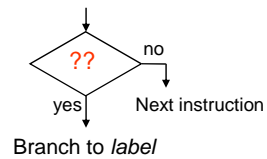
- Unconditional branches (BRA, BRN)

- Always branch to *label*



- Conditional branches (B??)

- Branch to *label* is taken if the condition ?? is satisfied. Otherwise, go to *next* instruction.



ELEC2220 Auburn University

3-4. Branch Instructions

- Conditions

- Single flag
N, Z, V, C
- Unsigned
Higher, Lower, Same
- Signed
Greater, Less, Equal

ELEC2220 Auburn University

3-4. Branch Instructions

3-4-2. Short and Long Branches

- Short (B??)
 - *Offset* is specified by a signed byte (8-bit)
 - $\$80$ (-128) \leq *Offset* \leq $\$7F$ (127)
 - $PC \leftarrow PC + \text{Offset}$
- Long (LB??)
 - *Offset* is specified by a signed word (16-bit)
 - $\$8000$ (-32768) \leq *Offset* \leq $\$7FFF$ (32767)
 - $PC \leftarrow PC + \text{Offset}$
 - Add “L” in front of each short branch instruction.
e.g., BRA \rightarrow LBRA, BGE \rightarrow LBGE

ELEC2220 Auburn University

3-4. Branch Instructions

Short
Branches

Unary Branches			
Mnemonic	Function	Equation or Operation	
BRA	Branch Always	$1 = 1$	
BRN	Branch Never	$1 = 0$	
Simple Branches			
Mnemonic	Function	Equation or Operation	
BCC	Branch if Carry Clear	$C = 0$	
BCS	Branch if Carry Set	$C = 1$	
BEQ	Branch if Equal	$Z = 1$	
BMI	Branch if Minus	$N = 1$	
BNE	Branch if Not Equal	$Z = 0$	
BPL	Branch if Plus	$N = 0$	
BVC	Branch if Overflow Clear	$V = 0$	
BVS	Branch if Overflow Set	$V = 1$	
Unsigned Branches			
Mnemonic	Function	Relation	Equation or Operation
BHI	Branch if Higher	$R > M$	$C + Z = 0$
BHS	Branch if Higher or Same	$R \geq M$	$C = 0$
BLO	Branch if Lower	$R < M$	$C = 1$
BLS	Branch if Lower or Same	$R \leq M$	$C + Z = 1$
Signed Branches			
Mnemonic	Function	Relation	Equation or Operation
BGE	Branch if Greater than or Equal	$R \geq M$	$N \oplus V = 0$
BGT	Branch if Greater Than	$R > M$	$Z + (N \oplus V) = 0$
BLE	Branch if Less than or Equal	$R \leq M$	$Z + (N \oplus V) = 1$
BLT	Branch if Less Than	$R < M$	$N \oplus V = 1$

ELEC2220 Auburn University

3-4. Branch Instructions

Long
Branches

Unary Branches		
Mnemonic	Function	Equation or Operation
LBRA	Long Branch Always	$1 = 1$
LBRN	Long Branch Never	$1 = 0$
Simple Branches		
Mnemonic	Function	Equation or Operation
LBCC	Long Branch If Carry Clear	$C = 0$
LBCS	Long Branch If Carry Set	$C = 1$
LBECQ	Long Branch If Equal	$Z = 1$
LBMI	Long Branch If Minus	$N = 1$
LBNE	Long Branch If Not Equal	$Z = 0$
LBPL	Long Branch If Plus	$N = 0$
LBVC	Long Branch If Overflow Clear	$V = 0$
LBVS	Long Branch If Overflow Set	$V = 1$
Unsigned Branches		
Mnemonic	Function	Equation or Operation
LBHI	Long Branch If Higher	$C + Z = 0$
LBHS	Long Branch If Higher Or Same	$C = 0$
LBLO	Long Branch If Lower	$Z = 1$
LBSL	Long Branch If Lower Or Same	$C + Z = 1$
Signed Branches		
Mnemonic	Function	Equation or Operation
LBGE	Long Branch If Greater Than Or Equal	$N \oplus V = 0$
LBGT	Long Branch If Greater Than	$Z + (N \oplus V) = 0$
LBLE	Long Branch If Less Than Or Equal	$Z + (N \oplus V) = 1$
LBLT	Long Branch If Less Than	$N \oplus V = 1$

ELEC2220 Auburn University

3-4. Branch Instructions

3-4-3. Bit Condition Branches

- Branch if bits in a memory byte are in a specific state.
 - A mask is used to test the byte.
 - If all bits in that byte corresponding to 1's in the mask are set (BRSET) or cleared (BRCLR), the branch is taken.
 - *Offset* is specified by a signed byte.
i.e., Short branch

Mnemonic	Function	Equation or Operation
BRCLR	Branch if Selected Bits Clear	$(M) \bullet (mm) = 0$
BRSET	Branch if Selected Bits Set	$(\bar{M}) \bullet (mm) = 0$

ELEC2220 Auburn University

3-4. Branch Instructions

3-4-3. Loop Primitive Instructions (DB??, IB??, TB??)

- The value of a counter is tested for *zero* or *not zero* (branch condition) with or without adjustment (+/- 1).
- Branch is taken if the condition is satisfied.
- Counter: A, B, D, X, Y, SP
- Offset is specified by a signed byte.
i.e., Short branch

ELEC2220 Auburn University

3-4. Branch Instructions

• Loop Primitive Instructions

Mnemonic	Function	Equation or Operation
DBEQ	Decrement Counter and Branch if = 0 (counter = A, B, D, X, Y, or SP)	$(\text{counter}) - 1 \Rightarrow \text{counter}$ If (counter) = 0, then Branch else Continue to next instruction
DBNE	Decrement Counter and Branch if \neq 0 (counter = A, B, D, X, Y, or SP)	$(\text{counter}) - 1 \Rightarrow \text{counter}$ if (counter) not = 0, then Branch else Continue to next instruction
IBEQ	Increment Counter and Branch if = 0 (counter = A, B, D, X, Y, or SP)	$(\text{counter}) + 1 \Rightarrow \text{counter}$ If (counter) = 0, then Branch else Continue to next instruction
IBNE	Increment Counter and Branch if \neq 0 (counter = A, B, D, X, Y, or SP)	$(\text{counter}) + 1 \Rightarrow \text{counter}$ if (counter) not = 0, then Branch else Continue to next instruction
TBEQ	Test Counter and Branch if = 0 (counter = A, B, D, X, Y, or SP)	If (counter) = 0, then Branch else Continue to next instruction
TBNE	Test Counter and Branch if \neq 0 (counter = A, B, D, X, Y, or SP)	If (counter) not = 0, then Branch else Continue to next instruction

ELEC2220 Auburn University

3-4. Branch Instructions

- **IF-THEN-ELSE** Construct

if a condition *then*
do this
else
do this

- **REPEAT-UNTIL** Construct

Do this
Until a condition

- **DO-WHILE** Construct

while a condition
do this

ELEC2220 Auburn University

3-5. Logic and Shift/Rotate Instructions

3-5-1. Logic

- Perform a bit-wise boolean logic operation between a register *r* (A,B, CCR) and a memory operand.
- Result saved in the register
- Three logic operations AND (AND*r*), OR (OR*r*), XOR (EOR*r*)
- Flags
 - N, Z, 0 → V for AND & OR
 - N, Z, V, C when CCR is involved.

ELEC2220 Auburn University

3-5. Logic and Shift/Rotate Instructions

- Logic

Mnemonic	Function	Operation
ANDA	AND A With Memory	$(A) \bullet (M) \Rightarrow A$
ANDB	AND B With Memory	$(B) \bullet (M) \Rightarrow B$
ANDCC	AND CCR With Memory (Clear CCR bits)	$(CCR) \bullet (M) \Rightarrow CCR$
EORA	Exclusive OR A With Memory	$(A) \oplus (M) \Rightarrow A$
EORB	Exclusive OR B With Memory	$(B) \oplus (M) \Rightarrow B$
ORAA	OR A With Memory	$(A) + (M) \Rightarrow A$
ORAB	OR B With Memory	$(B) + (M) \Rightarrow B$
ORCC	OR CCR With Memory (Set CCR bits)	$(CCR) + (M) \Rightarrow CCR$

ELEC2220 Auburn University

3-5. Logic and Shift/Rotate Instructions

3-5-2. Bit Test and Manipulation

- BIT r
 - Perform AND operation between a register r (A,B) and a memory operand, and the result is not saved.
- BSET
 - Set bits in a memory operand by performing OR operation between a memory operand and a mask.
 - The bits to be set is indicated by setting the corresponding bits in the mask.
- BCLR
 - Clear bits in a memory operand by performing AND operation between a memory operand and a mask.
 - The bits to be cleared is indicated by setting the corresponding bits in the mask.
- N, Z, 0 \rightarrow V

ELEC2220 Auburn University

3-5. Logic and Shift/Rotate Instructions

- Bit Test and Manipulation

Mnemonic	Function	Operation
BCLR	Clear Bits in Memory	$(M) \bullet (\overline{mm}) \Rightarrow M$
BITA	Bit Test A	$(A) \bullet (M)$
BITB	Bit Test B	$(B) \bullet (M)$
BSET	Set Bits In Memory	$(M) + (mm) \Rightarrow M$

ELEC2220 Auburn University

3-5. Logic and Shift/Rotate Instructions

3-5-3. Shift

- Logical Shift (LSL, LSL r , LSR r)
 - Shift a register r (A,B,D) or a memory operand to the left or right *one bit*.
- Arithmetic Shift (ASL, ASL r , ASR r)
 - Left: multiplying by 2
 - Right: dividing by 2 (MSB is replicated)
- N, Z, V, C

3-5-4. Rotate (ROL, ROR, ROL r , ROR r)

- Rotate a register r (A,B) or a memory operand to the left or right *through carry (C) one bit*.
- N, Z, V, C

ELEC2220 Auburn University

3-5. Logic and Shift/Rotate Instructions

- Shift and Rotate

Logical Shifts		
Mnemonic	Function	Operation
LSL LSLA LSLB	Logic Shift Left Memory Logic Shift Left A Logic Shift Left B	
LSLD	Logic Shift Left D	
LSR LSRA LSRB	Logic Shift Right Memory Logic Shift Right A Logic Shift Right B	
LSRD	Logic Shift Right D	

ELEC2220 Auburn University

3-5. Logic and Shift/Rotate Instructions

- Shift and Rotate

Arithmetic Shifts		
Mnemonic	Function	Operation
ASL ASLA ASLB	Arithmetic Shift Left Memory Arithmetic Shift Left A Arithmetic Shift Left B	
ASLD	Arithmetic Shift Left D	
ASR ASRA ASRB	Arithmetic Shift Right Memory Arithmetic Shift Right A Arithmetic Shift Right B	
Rotates		
Mnemonic	Function	Operation
ROL ROLA ROLB	Rotate Left Memory Through Carry Rotate Left A Through Carry Rotate Left B Through Carry	
ROR RORA RORB	Rotate Right Memory Through Carry Rotate Right A Through Carry Rotate Right B Through Carry	

ELEC2220 Auburn University