M68k Arithmetic Instructions

ECE 511: Digital System & Microprocessor

What we are going to learn in this session:

- M68k arithmetic instructions:
  - Plus
  - Minus
  - Multiply
  - Divide
  - Compare
  - Test
  - Negate

Introduction

Introduction

- M68k has instructions for simple arithmetic operations.
- Enhances ability to perform tasks:
  - Perform calculations + control tasks.
- Can also do floating-point, but requires math co-processor:
  - Not covered.

Some Basics

Addressing Modes

We have covered these addressing modes

Dn and An

DRD	D0 ➔ D7
ARD	A0 ➔ A7

Immediate data: <id>

ID #($/%/@/”)

Effective Address: <ea>

ARI	(An)
ARI+PI	(An)+
ARI+PD	-(An)
ARI+D	D(An)
ARI+I	D(An,Dn/An.s)
PC+D	D(PC)
PC+I	D(PC,Dn/An.s)
ALA	$001001
ASA	$FFAA
IA	CCR, SR, PC

Addressing Modes

- Dn: Data Register (D0 ➔ D7)
  - 32-bit register (32-bit max).
- An: Address Register (A0 ➔ A7).
  - 32-bit register (24-bit max).
  - Don’t use A7.

Addressing Modes

- <ea> Effective address (24-bit value):
  - $000000 ➔ $FFFFFF
  - Also includes Address Register Indirect methods.
  - Anything else: address error.
- <id> (Immediate Data):
  - Preceded by #.
  - Binary: %, Octal: @, Hex: $, Decimal: no sign, Character: ‘ ‘.

Addressing Modes Example

- D1, D2, D3, D4, …
- $123456
- 24(A0,D0.W)
- $123(PC)
- (A4)

- A1, A2, A3, A4, …
- #1000
- #$1F34
- #@4567
- #$00011011
- #’ABCD’

2’s Complement

2’s Complement

- Used by M68k to represent negative numbers.
- MSB as sign bit.
  - If 0, positive number.
  - If 1, negative number.

2's Complement Example

- Converting 10 to -10:

1. Start with positive number

10 (decimal) = 00001010 (binary)

2. Invert all the bits

$$\text{(invert)} \longrightarrow \begin{array}{r} 00001010 \\ \hline 11110101 \end{array}$$

3. Add 1 to inverted result

$$\begin{array}{rr} & 11110101 \\ + & 1 \\ \hline & 11110110 \end{array} \longleftarrow \text{2's Complement (-10)}$$

2's Complement Example

- Converting -5 to 5:

1. The 2's complement representation:

-5 (decimal) = 11111011 (binary)

2. Invert all the bits

$$\text{(invert)} \longrightarrow \begin{array}{r} 11111011 \\ \hline 00000100 \end{array}$$

3. Add 1 to inverted result

$$\begin{array}{rr} & 00000100 \\ + & 1 \\ \hline & 00000101 \end{array} \longleftarrow \text{Positive value (+5)}$$

Condition Code Register

Condition Code Register (CCR)

- Used to store status of evaluated conditions.
- Final 5-bits of SR.
- CCR = XNZVC
  - X: rotate, multi-precision BCD operations.
  - N: result is negative
  - Z: result is zero
  - V: overflow has occurred.
  - C: carry/borrow has occurred.

X Example – Rotate

ROXL.B #1,D0

D0.B =

1	0	0	1	1	1	0	0

***X keeps extra bit**

1	0	0	1	1	1	0	0

X

And moves it to the back…

0	0	1	1	1	0	0	1

X = 1 **C = 1** **C set as well**

N Example

- MOVE.B #0,D0
- MOVE.B #100,D1
- SUB.B D1,D0

D0 = $00 (0)
D1 = $64 (100)
D0 = 0 – 100 = $9C (-100)

$9C =

1	0	0	1	1	1	0	0

MSB = 1, N = 1

Z Example

- MOVE.L #$FFFFFFFF,D0
- SUB.B #$FF,D0

Initial D0 =	F	F	F	F	F	F	F	F
Final D0 =	F	F	F	F	F	F	0	0

Z = 1

*** Only B is tested, since .B was used**

V- Overflow Bit

- Set during arithmetic/divide overflow.
- For ADD, SUB & CMP:
  - P + P = N
  - N + N = P
  - P – N = N
  - N – P = P

  * P = Positive Number (MSB = 0)
  N = Negative Number (MSB = 1)
- More on divide overflow later.

V Example

- MOVE.B #$41,D0
- MOVE.B #$46,D1
- ADD.B D1,D0

D0 = 65 + 70

2's complement

	D0.B =	0	1	0	0	0	0	0	1	+65
+	D1.B =	0	1	0	0	0	1	1	0	+70
	New D0.B =	1	0	0	0	0	1	1	1	-121?

V = 1

V Example

- MOVE.B #$41,D0
- MOVE.B #$BA,D1
- SUB.B D1,D0

D0 = 65 – (-70)

2's complement

	D0.B =	0	1	0	0	0	0	0	1	+65
-	D1.B =	1	0	1	1	1	0	1	0	-70
	New D0.B =	1	0	0	0	0	1	1	1	-79?

V = 1

C Example - Carry

- MOVE.B #$FF,D0
- MOVE.B #$FF,D1
- ADD.B D1,D0

	D0.B =	1	1	1	1	1	1	1	1
+	D1.B =	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	0

Carry occurred, C = 1

C Example - Borrow

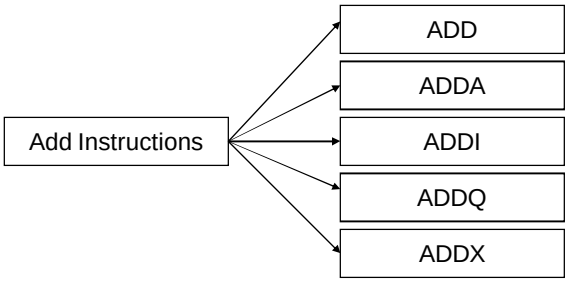
- MOVE.B #$12,D0
- MOVE.B #$34,D1
- SUB.B D1,D0

	D0.B =	0	0	0	1	0	0	1	0
-	D1.B =	0	0	1	1	0	1	0	0
		1	1	0	1	1	1	1	0

Borrow occurred, C = 1

Add Instructions

Available Instructions



ADD

- Adds two numbers together.
- D = S + D.
- Can use BWL.
- Effects all CCR.

ADD

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	s	s	s	s	s	s	s	s	s	s	s
d	d	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

How ADD Effects CCR

- X = set according to C.
- N = 1 if MSB is 1.
- Z = 1 if all active bits are 0.
- V = 1 if overflow occurred (4 rules).
- C = 1 if carry occurred.

ADD Example

- D0 = $000011FF
- D1 = $22223333
- ADD.B D0,D1

	D1 =	2	2	2	2	3	3	3	3
+	D0 =	0	0	0	0	1	1	F	F
	D1 =	2	2	2	2	3	3	3	2

Only lower byte changed, the rest are the same.

CCR
X = C
N = 0, MSB = 0
Z = 0, result non-zero.
V = 0, (N + P = P).
C = 1, carry occurred.

Try It Yourself

```
START   ORG     $1000

        move.l  #$000011ff,d0
        move.l  #$22223333,d1
        add.b   d0,d1

        END     START
```

ADDA (Add Address)

- Perform add to An:
  - Destination must be An.
- Can use WL:
  - W sign-extended to 32-bits before add.
- Doesn't effect CCR.

ADDA

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	s	s	s	s	s	s	s	s	s	s	s
-	d	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	-	-	-	-

WL

ADDA Example

D0 = $0000AAAA
A1 = $00001111
ADDA.W D0,A1

A0 =	0	0	0	0	1	1	1	1	
+	0	0	0	0	A	A	A	A	
A0 =	F	F	F	F	B	B	B	B	sign-extended.

X = unchanged
N = unchanged
Z = unchanged
V = unchanged
C = unchanged

Try It Yourself

```
START   ORG     $1000

        movea.l #$0000aaaa,a0
        movea.l #$00001111,a1
        adda.w  a0,a1

  END   START
```

ADDA Example

D0 = $0000AAAA
A1 = $00001111
ADDA.L D0,A1

A0 =	0	0	0	0	1	1	1	1
+ D0 =	0	0	0	0	A	A	A	A
A0 =	0	0	0	0	B	B	B	B

.L is used, value in D0 not sign-extended.

X = unchanged
N = unchanged
Z = unchanged
V = unchanged
C = unchanged

Try It Yourself

```
START   ORG     $1000

        movea.l #$0000aaaa,a0
        movea.l #$00001111,a1
        adda.l  a0,a1

  END   START
```

ADDI (Add Immediate)

- Adds immediate data to destination.
- Source must be immediate data.
- Can use BWL.
- Effects all CCR (similar to ADD).

ADDI

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

ADDI Example

D0 = $12345678
ADDI.L #$44,D0

D0=	1	2	3	4	5	6	7	8
+	0	0	0	0	0	0	4	4
D0=	1	2	3	4	5	6	B	C

CCR
X = Set similar to C
N = 0 (MSB = 0)
Z = 0 (result nonzero)
V = 0 (P + P = P)
C = 0 (No carry)

Try It Yourself

```
START   ORG     $1000

        move.l  #$12345678,d0
        addi.l  #$44,d0

  END   START
```

ADDQ (Add Quick)

- Similar to ADDI, but immediate data between 1 ➔ 8.
- Can use BWL.
- Effects all CCR (similar to ADD).
- Generates smaller MC, faster execution.

ADDQ

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	d	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

ADDQ Example

D0 = $12345678
ADDQ.B #1,D0

D0=	1	2	3	4	5	6	7	8
+							0	1
D0=	1	2	3	4	5	6	7	9

CCR
X = Set similar to C
N = 0 (MSB = 0)
Z = 0 (result nonzero)
V = 0 (P + P = P)
C = 0 (No carry)

Try It Yourself

```
START   ORG     $1000

        move.l  #$12345678,d0
        addq.l  #$01,d0

  END   START
```

This Causes an Error. Why?

```
START   ORG     $1000

        move.l  #$12345678,d0
        addq.l  #$9,d0

        END     START
```

ADDX (Add Extended)

- Adds X together with results.
- D = D + S + X.
- Can use BWL.
- Limited addressing modes.
- Effects all CCR.

ADDX

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	s	-	-	-	-	-	-	-
d	-	-	-	d	-	-	-	-	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

ADDX Example

D0 = $12345678
D1 = $ABCDEF12
X = 1
ADDX.B D1,D0

D0=	1	2	3	4	5	6	7	8
D1=	A	B	C	D	E	F	1	2
+								1
D0=	1	2	3	4	5	6	8	B

<u>CCR</u>
X = Set similar to C
N = 1 (MSB = 1)
Z = 0 (result nonzero)
V = 1 (P + P = N)
C = 0 (No carry)

Try It Yourself

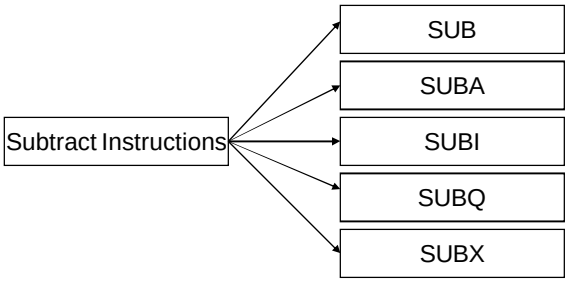
```
START    ORG      $1000

         move.l   #$12345678,d0
         move.l   #$ABCDEF12,d1
         and.b    #$00,ccr
         or.b     #$10,ccr
         addx.b   d1,d0

         END      START
```

Subtraction Instructions

Available Instructions



SUB

- Subtracts source from destination.
- D = D – S.
- Can use BWL.
- Effects all CCR.

SUB

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	s	s	s	s	s	s	s	s	s	s	s
d	d	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

How SUB Effects CCR

- X = set according to C.
- N = 1 if MSB is 1.
- Z = 1 if all active bits are 0.
- V = 1 if overflow occurred.
- C = 1 if borrow occurred.

SUB Example

- D0 = $00004444
- D1 = $22223333
- SUB.W D0,D1

	D1 =	2	2	2	2	3	3	3	3
-	D0 =	0	0	0	0	4	4	4	4
(destination)	D1 =	2	2	2	2	E	E	E	F

X = C
N = 1, MSB = 1
Z = 0, result non-zero.
V = 0 (P – P = N).
C = 1, borrow occurred.

Try It Yourself

```
START   ORG     $1000

        move.l  #$00004444,d0
        move.l  #$22223333,d1
        sub.w   d0,d1

        END     START
```

SUBA (Subtract Address)

- Used to subtract values from address register.
- Can use WL.
- Source not sign-extended (unlike ADDA).
- Doesn't effect CCR.

SUBA

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	s	s	s	s	s	s	s	s	s	s	s
-	d	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	-	-	-	-

WL

SUBA Example

- A0 = $00002222
- D1 = $2222EEEE
- SUBA.W D1,A0

	A0 =	0	0	0	0	2	2	2	2
-	D1 =	2	2	2	2	E	E	E	E
(destination)	A0 =	0	0	0	0	3	3	3	4

XNZVC unchanged

Only lower word changed, the rest are the same.

Try It Yourself

```
START   ORG     $1000

        move.l  #$00002222,a0
        move.l  #$2222EEEE,d1
        suba.w  d1,a0

        END     START
```

SUBI (Subtract Immediate)

- Subtracts immediate data from destination.
- Can use BWL.
- Effects all CCR similar to SUB.

SUBI

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

SUBI Example

- D0 = $00001234
- SUBI.W #$3345,D0

D1 =	0	0	0	0	1	2	3	4
-	0	0	0	0	3	3	4	5
(destination) D1 =	0	0	0	0	D	E	E	F

Only lower word changed, the rest are the same.

X = C
N = 1, MSB = 1
Z = 0, result non-zero.
V = 0 (P – P = N).
C = 1, borrow occurred.

Try It Yourself

```
START   ORG     $1000

        move.l  #$00001234,d0
        SUBI.W  #$3345,D0

        END     START
```

SUBQ (Subtract Quick)

- Similar to SUBI, but immediate data between 1 ➔ 8.
- Source must be immediate data.
- Can use BWL.
- Effects all CCR (similar to SUB).
- Generates smaller MC, faster execution.

SUBQ

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

SUBQ Example

D0 = $12345678
SUBQ.B #$07,D0

D0=	1	2	3	4	5	6	7	8
-							0	7
D0=	1	2	3	4	5	6	7	1

CCR
X = Set similar to C
N = 0 (MSB = 0)
Z = 0 (result nonzero)
V = 0 (P - P = P)
C = 0 (No carry)

Try It Yourself

```
START   ORG     $1000

        move.l  #$12345678,d0
        SUBQ.B   #$07,D0

        END     START
```

SUBX (Subtract Extended)

- Subtracts X with results.
- D = D - S - X.
- Can use BWL.
- Limited addressing modes.
- Effects all CCR.

SUBX

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	-	-	s	-	-	-	-	-	-	-
d	-	-	-	d	-	-	-	-	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

SUBX Example

D0 = $12345678
D1 = $ABCDEFAB
X = 1
SUBX.B D1,D0

D0=	1	2	3	4	5	6	7	8
D1=	A	B	C	D	E	F	A	B
-								1
D0=	1	2	3	4	5	6	C	C

<u>CCR</u>
X = Set similar to C
N = 1 (MSB = 1)
Z = 0 (result nonzero)
V = 1 (P – N = N)
C = 1 (borrow occurred)

Try It Yourself

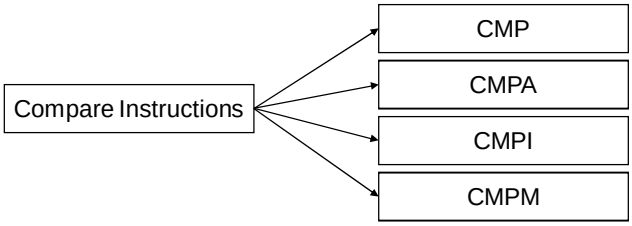
```
START    ORG      $1000

         MOVE.L   #$12345678,D0
         MOVE.L   #$ABCDEFAB,D1
         AND.B    #$00,CCR
         OR.B     #$10,CCR
         SUBX.B    D1,D0

         END  START
```

Compare Instruction

Available Instructions



CMP (Compare)

- Compare destination to source:
  - Similar to SUB (D - S), but doesn't modify source/destination.
  - Only modifies CCR.
  - Destination must be data register.
- Can use BWL.
- Effects all CCR except X.

CMP vs. SUB

	SUB	CMP
Method of operation	Subtracts source from destination.	Subtracts source from destination.
Valid operand size	BWL	BWL
CCR Result	Updated according to subtract operation. Effects all CCR.	Updated according to “subtract” operation. Effects all CCR except X.
Effect on X	X = C	X unchanged.
Final result	Result stored in destination.	Result not stored, only CCR updated.

CMP

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	s	s	s	s	s	s	s	s	s	s	s
d	-	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	*	*

BWL

How CMP Effects CCR

- X = unchanged.
- N = 1 if MSB is 1.
- Z = 1 if all active bits are 0.
- V = 1 if overflow occurred.
- C = 1 if borrow occurred.

CMP Example

- D0 = $00001212
- D1 = $22221234
- CMP.B D0,D1

D1 =	2	2	2	2	1	2	3	4
- D0 =	0	0	0	0	1	2	1	2
"D1" =	2	2	2	2	1	2	2	2
Actual D1 =	2	2	2	2	1	2	3	4

X = unchanged
N = 0 (MSB = 0)
Z = 0 (result non-zero)
V = 0 (P – P = P).
C = 0 (no borrow).

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$00001212,D0
        MOVE.L  #$22221234,D1
        CMP.B   D0,D1

        END     START
```

CMP Example

D3 =

0	1	2	3	4	5	6	7

```
CMP.W     #$4567,D3

D3.W =    $4567
-         $4567
----------------
          $0000
```

Final D3 =

0	1	2	3	4	5	6	7

*Original D3 value unchanged.

X = unchanged
N = 0 (MSB = 0)
Z = 1 (result zero)
V = 0 (P – P = P).
C = 0 (no borrow).

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$01234567,D3
        CMP.W   #$4567,D3

        END     START
```

CMP Example

D3 =

0	0	0	0	0	0	5	0

D4 =

0	0	0	0	0	0	6	0

CMP.B D4,D3

D3.B =		$50
D4.B =	-	$60
		$F0

Final D3 =

0	0	0	0	0	0	5	0

Final D4 =

0	0	0	0	0	0	6	0

*Original D3, D4 unchanged.

X = unchanged
N = 1 (MSB = 1)
Z = 0 (result nonzero)
V = 0 (P – P = N).
C = 1 (borrow occurred).

CMPA (Compare Address)

- Compare address register to source:
  - Doesn't modify source/destination.
  - Only modifies CCR.
  - Destination must be address register.
- Can use WL.
- Effects all CCR except X.

CMPA

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	s	s	s	s	s	s	s	s	s	s	s
-	d	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	*	*

WL

CMPA Example

- A0 = $00002222
- D1 = $2222EEEE
- CMPA.W D1,A0

	A0 =	0	0	0	0	2	2	2	2
-	D1 =	2	2	2	2	E	E	E	E
	"A0" =	0	0	0	0	3	3	3	4
	Final A0 =	0	0	0	0	2	2	2	2

X = unchanged
N = 0 (MSB is 0)
Z = 0 (result nonzero)
V = 0 (P – N = P)
C = 1 (borrow occurred)

Try It Yourself

```
START   ORG     $1000

        MOVEA.L #$00002222,A0
        MOVE.L  #$2222EEEE,D1
        CMPA.W  D1,A0

        END     START
```

CMPI (Compare Immediate)

- Compares destination to immediate data.
- Source must be immediate data.
- Can use BWL.
- Effects all CCR except X (same as CMP).

CMPI

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	-	-	-	-	-	-	-	-	s
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	*	*

BWL

CMPI Example

- D4 = $0000AABB
- CMPI.W #$7ABB,D4

D4 =	0	0	0	0	A	A	B	B
-	0	0	0	0	7	A	B	B
"D4" =	0	0	0	0	3	0	0	0
Actual D4 =	0	0	0	0	A	A	B	B

X = unchanged
N = 0, MSB = 0
Z = 0, result non-zero.
V = 1 (N – P = P).
C = 0, no borrow.

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$0000AABB,D4
        CMPI.W  #$7ABB,D4

        END     START
```

CMPM (Compare Memory)

- Compare between two memory locations.
- Can use BWL.
- Source & destination must be memory.
- Effects all CCR except X (same as CMP).
- Only ARI + PI addressing allowed.

CMPM

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
-	-	-	s	-	-	-	-	-	-	-	-
-	-	-	d	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	*	*

BWL

CMPM Example

A0 = $001002
A1 = $002000
CMPM.W (A0)+,(A1)+

$1000	00
$1001	11
$1002	22
$1003	33
$1004	44

$2000	AA
$2001	11
$2002	33
$2003	66
$2004	76

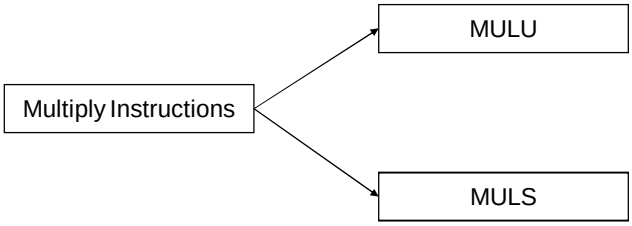
(A1) =	A	A	1	1
- (A0) =	2	2	3	3
Result =	8	7	D	E

*Memory values unchanged.
* A1, A0 increased by 2

X = unchanged
N = 1 (MSB = 1)
Z = 0 (Result non-zero)
V = 0 (N – P = N)
C = 0 (No carry)

Multiply Instructions

Available Instructions



MULU (Multiply Unsigned)

- Multiplies two 16-bit unsigned numbers and produces 32-bit unsigned result.
- Destination MUST be Dn.
- Effects all CCR except X.

MULU

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	s	s	s	s	s	s	s	s	s	s
d	-	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	0	0

W x W

MULU Example

D0 = | 0 | 0 | 0 | 0 | 0 | 0 | 6 | 4 | 100 decimal

D1 = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 2 decimal

MULU D1,D0

16-bit 16-bit

0	0	6	4	X	0	0	0	2

100 x 2 = 200 = $C8

D0 = | 0 | 0 | 0 | 0 | 0 | 0 | C | 8 |

D0 replaced completely

X = unchanged
N = 0 (MSB = 0)
Z = 0 (result non-zero)
V = 0
C = 0

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$64,D0
        MOVE.L  #$02,D1
        MULU    D1,D0

        END     START
```

MULS (Multiply Signed)

- Multiplies two 16-bit signed numbers and produces 32-bit signed result.
- Destination MUST be Dn.
- Effects all CCR except X.

MULS

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	s	s	s	s	s	s	s	s	s	s
d	-	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	0	0

W x W

MULS Example

D0 = | 0 | 0 | 0 | 0 | F | E | 0 | C | -500 decimal

D1 = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 decimal

MULU D1,D0

16-bit 16-bit

F	E	0	C	X	0	0	0	5

-500 x 5 = -2500 = $F63C

D0 = | F | F | F | F | F | 6 | 3 | C |

D0 replaced completely

X = unchanged
N = 1 (MSB is 1)
Z = 0 (result non-zero)
V = always cleared
C = always cleared

Try It Yourself

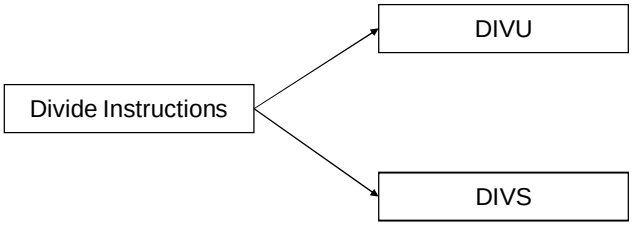
```
START   ORG     $1000

        MOVE.L  #$0000FE0C,D0
        MOVE.L  #$00000005,D1
        MULS    D1,D0

        END     START
```

Divide Instructions

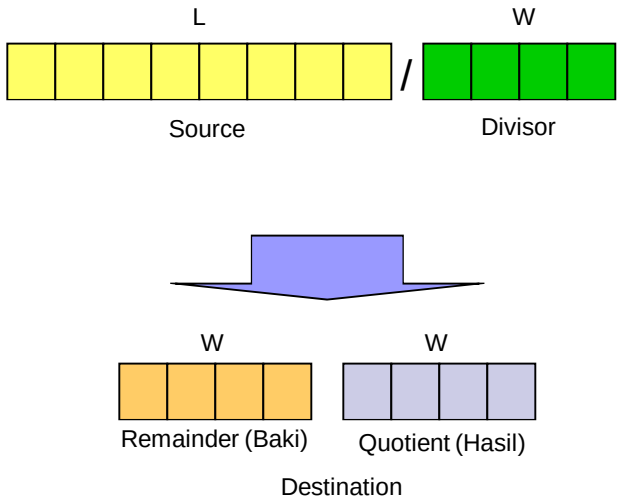
Available Instructions



DIVU (Divide Unsigned)

- Divides 32-bit unsigned number by 16-bit unsigned number.
- Produces 32-bit result.
  - Lower 16-bits are quotient (hasil).
  - Upper 16-bits are remainder (baki).
- Destination MUST be Dn.
- Effects all CCR except X.

DIVU



DIVU

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	s	s	s	s	s	s	s	s	s	s
d	-	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	*	0

L ÷ W

How DIVU Effects the CCR

- X = unchanged.
- N = Set if quotient MSB is 1.
- Z = Set if quotient is zero.
- V = Set if division overflow occurs.
  - V = undefined if divide-by-zero.
- C = Always cleared.

DIVU Example

D0 = $000186A1 (100,001 decimal)
D1 = $000000C (12 decimal)
DIVU D1,D0

100,001 / 12 = 8333, remainder 5

	Quotient	Remainder
Decimal	8,333	5
Hex	$208D	$0005

New D0 =

0	0	0	5	2	0	8	D

X = unchanged
N = 0 (Q MSB is 0)
Z = 0 (Q nonzero)
V = 0 (no overflow)
C = always cleared

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$000186A1,D0
        MOVE.L  #$0000000C,D1
        DIVU    D1,D0

        END     START
```

DIVU Example - Overflow

D0 = $00186A1 (100,001 decimal)
D1 = $0000001 (1 decimal)
DIVU D1,D0

100,001 / 1 = 100,001 = $186A1

	Quotient	Remainder
Decimal	100,001	0
Hex	$186A1	$0000

New D0 =

0	0	0	1	8	6	A	1

Quotient overflowed to remainder.

X = unchanged
N = 0 (MSB is 0)
Z = 0 (result nonzero)
V = 1 (divide overflow)
C = always cleared

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$000186A1,D0
        MOVE.L  #$00000001,D1
        DIVU    D1,D0

        END     START
```

DIVS (Divide Signed)

- Divides 32-bit signed number by 16-bit signed number.
- Division method same as DIVU.
- Destination MUST be Dn.
- Effects CCR similar to DIVU.

DIVS

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
s	-	s	s	s	s	s	s	s	s	s	s
d	-	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	*	0

L ÷ W

DIVS Example

D0 = #$00001389 (5,001)
D1 = #$0000FFE7 (-25)

DIVS D1,D0 5,001 / (-25) = -200, remainder 1

	Quotient	Remainder
Decimal	-200	1
Hex	$FF38	$0001

New D0= 0 0 0 1 F F 3 8

X = unchanged
N = 1 (Q MSB is 1)
Z = 0 (Q result nonzero)
V = 0 (no overflow)
C = always cleared

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$00001389,D0
        MOVE.L  #$0000FFE7,D1
        DIVS    D1,D0

        END     START
```

Test Instructions

TST (Test Operand)

- Allows CCR to be updated based on current destination value:
  - Compares destination to zero.
  - Similar to CMP.s #0,<destination>
- Destination not effected:
  - Only changes CCR.
- Can use BWL.
- Effects all CCR except X.

TST

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

BWL

TST Example

D3 =

F	2	3	4	5	6	7	8

TST.L D3 (TST.L D3 = CMP.L #0,D3)

D3.L = $F2345678
\- $00000000

$F2345678

Final D3 =

F	2	3	4	5	6	7	8

*D3 unchanged, CCR updated.

X = unchanged
N = 1 (MSB is 1)
Z = 0 (data nonzero)
V = always cleared
C = always cleared

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$F2345678,D3
        TST.L   D3

        END     START
```

TAS (Test & Set Operand)

- Similar to TST, but only tests 1 byte:
  - Tests byte and sets CCR.
  - Then set MSB to 1.
- Can only use B.
- Effects all CCR except X.

TAS

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
-	*	*	0	0

B

TAS Example

Only byte tested

D3 = | F | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

TAS D3

1 D3.B = $78
- $00

$78

X = unchanged
N = 0 (MSB is 0)
Z = 0 (result non-zero)
V = always cleared.
C = always cleared.

2 **After testing, bit 7 is set.**

D3.B = | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

F **8**

Final D3 = | F | 2 | 3 | 4 | 5 | 6 | F | 8 |

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$F2345678,D3
        TAS     D3

        END     START
```

Other Instructions

NEG (Negate)

- Converts value to opposite sign:
  - Negative ➔ Positive
  - Positive ➔ Negative
  - 0 – D = D
- 2's Complement representation.
- Can use BWL.
- Effects all CCR.

NEG

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	1

BWL

NEG Example

Find the 2's Complement value of -10 when D0 is loaded with 10.

D0 = $0000000A
NEG.B D0

00 =	0	0	0	0	0	0	0	0	0
- D0 =	0	0	0	0	0	0	0	A	-10
D0 =	0	0	0	0	0	0	F	6	-10

C is always 1 since the operation always requires a carry operation.

X = C
N = 1 (MSB is 1)
Z = 0 (result is nonzero)
V = 0 (P – P = N)
C = 1 (borrow occurred).

Try It Yourself

```
START   ORG     $1000

        MOVE.B  #10,D0
        NEG.B   D0

        END     START
```

What if we negate again?

D0 = $000000F6
NEG.B D0

00 =	0	0	0	0	0	0	0	0	0
- D0 =	0	0	0	0	0	0	F	6	-(-10)
D0 =	0	0	0	0	0	0	0	A	+10

X = C
N = 0 (MSB is 0)
Z = 0 (result is nonzero)
V = 0 (P – N = P)
C = 1 (borrow occurred).

Try It Yourself

```
START   ORG     $1000

        MOVE.B  #10,D0
        NEG.B   D0
        NEG.B   D0

        END     START
```

NEGX (Negate with Extend)

- Same as NEG, but subtracts X as well:
  - 0 – D – X = D
- 2's Complement representation.
- Can use BWL.
- Effects all CCR.

NEGX

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
d	-	d	d	d	d	d	d	d	-	-	-

X	N	Z	V	C
*	*	*	*	*

BWL

NEGX Example

D0 = $00000011
X = 1
NEGX.B D0

00 =	0	0	0	0	0	0	0	0	0
- D0 =	0	0	0	0	0	0	1	1	-17
- X =							0	1	-1
D0 =	0	0	0	0	0	0	E	E	-18

X = C
N = 1 (MSB is 1)
Z = 0 (result is nonzero)
V = 0 (P – P = N)
C = 1 (borrow occurred).

Try It Yourself

```
START    ORG      $1000

         MOVE.B   #$11,D0
         AND.B    #$00,CCR
         OR.B     #$10,CCR
         NEGX.B   D0

         END      START
```

EXT (Sign Extend)

- Extends sign to remaining upper register bits.
- Can use WL only.
- Checks MSB (sign bit) and extends sign:
  - If .W used, extended to 16-bits.
  - If .L used, extended to 32-bits.
- Effects all CCR except X.

EXT

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,i)	ASA	ALA	d(PC)	d(PC,i)	#n
d	-	-	-	-	-	-	-	-	-	-	-

X	N	Z	V	C
-	*	*	0	0

WL

EXT Example

MSB is 1

$A = 1010

D0 =

0	0	0	0	0	0	A	A

EXT.W D0

If .W used, check MSB of byte, then extend to word-length.

D0 =

0	0	0	0	F	F	A	A

X = unchanged
N = 1 (MSB is 1)
Z = 0 (result is nonzero)
V = always cleared
C = always cleared.

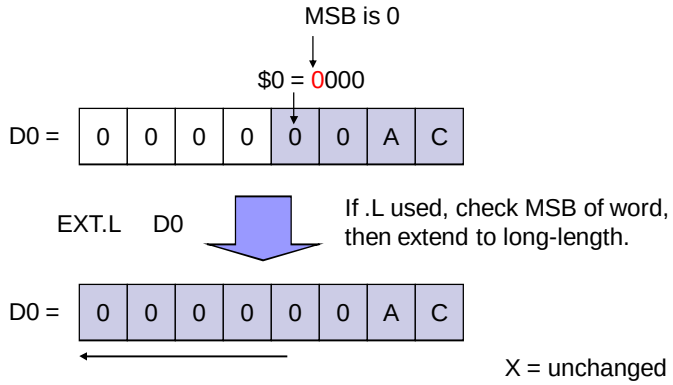
Try It Yourself

```
START   ORG     $1000

        MOVE.B  #$AA,D0
        EXT.W   D0

        END     START
```

EXT Example



X = unchanged
N = 0 (MSB is 0)
Z = 0 (result is nonzero)
V = always cleared
C = always cleared.

Try It Yourself

```
START   ORG     $1000

        MOVE.W  #$00AC,D0
        EXT.L   D0

        END     START
```

Important Note on EXT

- If .W is used, EXT checks MSB of byte and extends to word value.
- If .L is used EXT checks MSB of word and extends to long value.

Conclusion

Summary of ADD Instructions

Instruction	Purpose	Operation
ADD	Add together two numbers	D + S = D
ADDA	Add source to address register	An + S = D
ADDI	Add immediate data to destination	D + <id> = D
ADDQ	Similar to ADDI, but immediate data 1 ➔ 8	D + <id> = D
ADDX	Add together source, destination and X bit.	D + S + X = D

Summary of SUB Instructions

Instruction	Purpose	Operation
SUB	Subtracts source from destination	D - S = D
SUBA	Subtract source from address register	An - S = D
SUBI	Subtract immediate data from destination	D - <id> = D
SUBQ	Similar to SUBI, but immediate data 1 ➔ 8	D - <id> = D
SUBX	Subtract source and X bit from destination.	D - S - X = D

Summary of CMP Instructions

Instruction	Purpose	Operation
CMP	Similar to SUB, but doesn't modify D. Only updates CCR.	D - S = "D"
CMPA	Compare address register to source. Only updates CCR.	An - S = "D"
CMPI	Compare D to immediate data. Only updates CCR.	D - <id> = "D"
CMPM	Compare between 2 memory locations. Only updates CCR.	M1 – M2 = "D"

Summary of MUL & DIV Instructions

Instruction	Purpose	Operation
MULU	Multiples two unsigned W data and store in D	W x W = L
MULS	Multiplies two signed W data and store in D	W x W = L
DIVU	Divides unsigned L data with W data and store in D	L/W = W_Q,W_R
DIVS	Divides signed L data with D data and store in D	L/W = W_Q,W_R

Summary of Instructions

Instruction	Purpose	Operation
TST	Compare D to zero, and update CCR	CMP #0,D
TAS	Compare D_B with zero, update CCR, and set MSB	CMP #0,D MSB = 1
NEG	Negates D (2's complement)	D = 0 - S
NEGX	Negates D, then subtract X	D = 0 – S - X
EXT	Sign extends B ➔ W, W ➔ L	N/A

The End

Please read:

Antonakos, 69 - 76