M68k Addressing Modes

ECE 511: Digital System & Microprocessor

What we will learn in this session:

- M68k addressing modes:
  - How to access data:
    - In registers.
    - In memory.
  - Available addressing modes.
  - When to use what.

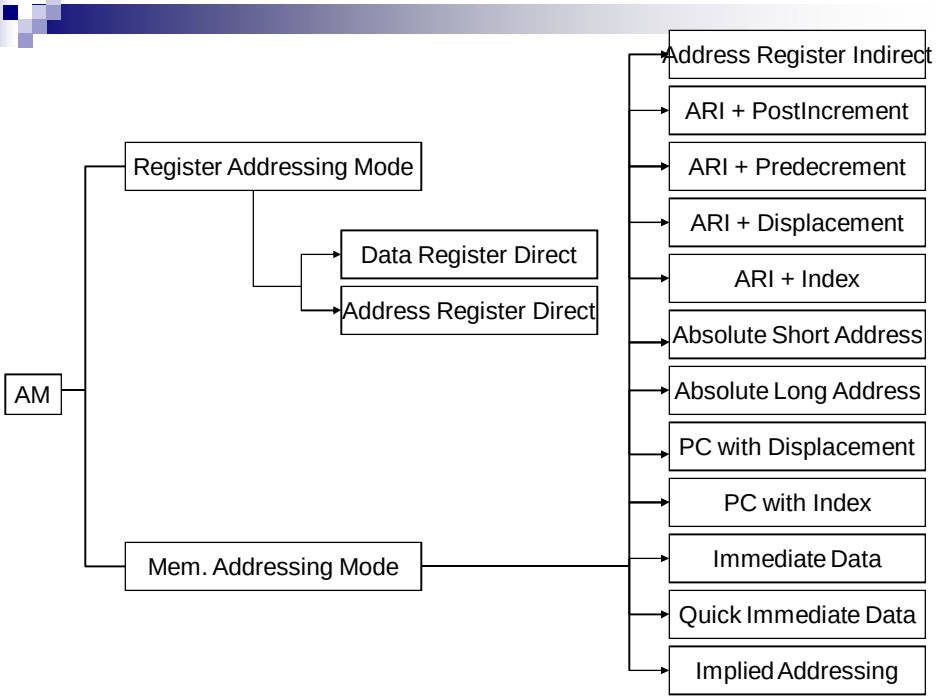
Introduction

Introduction

- All CPU instructions have similar requirements:
  - Action - What action should it perform?
  - Source - Where is the data that it is supposed to process?
  - Target - Where should it put the results?
- Action is done using instructions.
- Addressing modes identify source and target.

Addressing Modes

- Methods to access data inside:
  - CPU registers.
  - Memory.
- M68k allows 14 addressing modes:
  - Direct reference.
  - Indirect reference.
- Addressing modes allow flexible & effective programs design.



Register Addressing Modes

Register Addressing Modes

- Modes to access registers in M68k:
  - Address Register.
  - Data Register.
- Consists of 2 methods:
  - Data Register Direct (DRD).
  - Address Register Direct (ARD).

DRD (Data Register Direct)

- Used to access data registers.
- Represented to as D_n:
  - D represents data register.
  - n is register number.
  - From D_0 to D_7.

DRD Example

- MOVE.B D0,D1
- ADD.W D4,(A0)
- MULU D5,D7

ARD (Address Register Direct)

- Used to access address registers.
- Referred to as A_n:
  - A represents address register.
  - n is register number.
  - From A_0 to A_7.
- A_7 = stack pointer.

ARD Example

- MOVEA.L A0,A4
- ADDA.W A3,A6
- MOVEA.W #$1234,A2
- LEA $1000,A4

Memory Addressing Modes

Memory Addressing Modes

- Modes to access memory locations.
- 12/14:
  - Memory space is large area.
  - Many varieties of addressing modes.
  - Depends on desired function.

ARI (Address Register Indirect)

- Refers to contents of memory location *pointed* by A_n.
- Address register enclosed in parenthesis ➔ (A_n).

Example: ARI

- D0 = $12345678
- A1 = $007A92
- MOVE.B D0,(A1)

  (This command does not move the data inside D0 into A1, but moves the data inside D0 into the *memory location pointed* by A1).

Example: ARI

- MOVE.B D1,(A4)
  - Moves a byte from D1 into the memory location specified by A4.
- ADD.W (A3),D3
  - Adds the word content of memory address specified by A3 to data register D3.

Example: ARI

- D0 = $12345678
- A1 = $00007A92
- MOVE.L D0,(A1)

D0.L = $12345678

A1 = $007A92 (A1 is still unchanged).

Memory Contents

$7A90	
$7A91	
$7A92	$12
$7A93	$34
$7A94	$56
$7A95	$78

Try It Yourself

```
START   ORG     $1000

        MOVE.L  #$12345678,D0
        LEA     $7A92,A1
        MOVE.L  D0,(A1)

        END     START
```

ARI + PI (Address Register Indirect with Post-Increment)

- Same as ARI, but A_n automatically incremented after execution (post-increment).
- Use the '+' sign after (A_n).
- Useful in `for` loops.

ARI + PI (Address Register Indirect with Post-Increment)

- Increment value depends on data length:
  - If .B is used, A_n is incremented by 1.
  - If .W is used, A_n is incremented by 2.
  - If .L is used, A_n is incremented by 4.

Example 1: ARI + PI

- D0 = $12345678
- A1 = $001002
- MOVE.W D0,(A1)+

D0 = $12345678

Memory Content

$1001	
$1002	$56
$1003	$78
$1004	
$1005	
$1006	

After execution, A1 is incremented by 2 since .W was used.

A1 = $1002 + 2 = $001004 (new value).

Try It Yourself

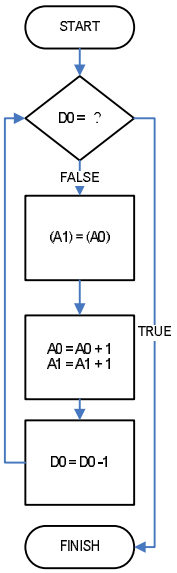
```
START   ORG     $1000

        MOVE.L  #$12345678,D0
        LEA     $1002,A1
        MOVE.W  D0,(A1)+

        END     START
```

Example 2: ARI + PI

D0 = $00000005 *as counter
A0 = $001000
A1 = $002000

```
LABEL1    MOVE.B    (A0)+,(A1)+
          SUB.B     #1,D0
          CMP.B     #0,D0
          BNE       LABEL1
```

D0	A0	A1
5	$1000	$2000
4	$1001	$2001
3	$1002	$2002
2	$1003	$2003
1	$1004	$2004
0	$1005	$2005



Try It Yourself

```
START       ORG         $1000

            MOVE.B      #5,D0
            LEA         $1000,A0
            LEA         $2000,A1

LABEL1      MOVE.B      (A0)+,(A1)+
            SUB.B       #1,D0
            CMP.B       #0,D0
            BNE         LABEL1

            END         START
```

ARI + PD (Address Register Indirect with Pre-Decrement)

- Same as ARI, but value in address register automatically decremented before execution (pre-decrement).
- Use the '-' before (A_n) sign.
- Useful to push data to stack.

ARI + PD (Address Register Indirect with Pre-Decrement)

- The increment value depends on data length:
  - If .B is used, A_n is decremented by 1.
  - If .W is used, A_n is decremented by 2.
  - If .L is used, A_n is decremented by 4.

Example: ARI + PD – Moving Data to Stack

- D0 = $12345678
- A6 = $001002
- MOVE.B D0,-(A6)

Before execution, A6 is decremented by 1 since .B was used.

A6 = $1002 - 1 = $001001 (new value).

D0 = $12345678

Memory Contents

A6 (SP)	$1001	$78
	$1002	
	$1003	
	$1004	
	$1005	
	$1006	

Try It Yourself

```
START   ORG     $2000

        MOVE.L  #$12345678,D0
        LEA     $1002,A6
        MOVE.B  D0,-(A6)

        END     START
```

ARI + D (Address Register Indirect with Displacement)

- Adds a displacement value to ARI.
- Format: $d(A_n)$
  - A_n is address register.
  - d is 16-bit displacement value.
- The range of displacement is from $0000 to $FFFF.

Example: ARI + D

D3 = $12345678

A4 = $004500

- MOVE.B D3,$750(A4)

A4=$004500, Disp.=$750

Effective Address:

A4	$004500	
D	$000750	+
EA	$004C50	

Memory Contents

$4C4C	
$4C4D	
$4C4E	
$4C4F	
$4C50	$78
$4C51	

Try It Yourself

```
START    ORG      $2000

         MOVE.L   #$12345678,D3
         LEA      $4500,A4

         MOVE.B   D3,$750(A4)
         LEA      $750(A4),A5

         END      START
```

Example: ARI + D

D3 = $00000000

A4 = $004500

- MOVE.B $FFF3(A4),D3

$FFF3 is negative (MSB = 1)
2's complement = $000D

Effective Address:

A4	$004500
Disp.	$00000D –
	$0044F3

Memory Contents

$44F1	$11
$44F2	$22
$44F3	$33
$44F4	$44
$44F5	$55
$44F6	$66

D3 = $00000033

Try It Yourself

```
START    ORG      $2000

         MOVE.L   #$12345678,D3
         LEA      $4500,A4

         MOVE.B   D3,$FFF3(A4)
         LEA      $FFF3(A4),A5

         END      START
```

Example: ARI + D

D3 = $00000000

A4 = $004500

- MOVE.B -5(A4),D3

Memory Contents

$44F9	
$44FA	
$44FB	$14
$44FC	
$44FD	
$44FE	

Effective Address:
A4 $004500
Disp. $000005 –
$0044FB

D3 = $00000014

ARI + D Example

- MOVE.B D1,34(A0)
- ADD.B $1254(A2),D4

•Displacement must not be more than 16-bits long.

ARI + I (Address Register Indirect with Index)

- Similar to ARI + D, but adds another index term.
- Displacement range $80 (-128) < D < $7F (127).
- Index term from Dn or An.
- Used for implementing 2-D arrays.
- Adds index term into bracket:
  - D(An,Dn.W/L)
  - D(An,An.W/L)
- Effective address is ARI + D + Index (Dn.W/L or An.W/L).

ARI + I Example

- MOVE.B D1,34(A0,D3.W)
- ADD.B $54(A2,A4.W),D4

•Displacement must be 8-bits long.
•Index must be 16-bits long.

ARI + I Example

D1 = $00000012
A4 = $00001000
MOVE.B #$FF,$78(A4,D1.W)

Effective Address (ARI + D + I):

	ARI	$	00001000
+	D1.W	$	0012
+	D	$	78
	EA	$	0000108A

$1088	
$1089	
$108A	$FF
$108B	
$108C	
$108D	
$108E	

Try It Yourself

```
START       ORG         $2000

            MOVE.W      #10,D0
            LEA         $5000,A0

* EFFECTIVE ADDRESS IS
* $5000 + $03 + $0A = $500D

            LEA         3(A0,D0.W),A1

            END         START
```

ALA (Absolute Long Address)

- Directly addresses memory locations.
- Address must be 24-bits.
- No sign extension performed.
- Slower than ASA, requires more machine code.

ALA Example

- MOVE.L D0,$100100
- ADD.B $001000,D3
- MOVE.B $000100,$400400

Example: ALA

- MOVE.B $001000,D0
  - Moves byte content of memory address $1000 to D0.
- ADD.W $400400,D1
  - Adds the word content of $400400 to D1.
- MOVE.L D4,$003000
  - Moves a long-word from D4 to address $3000.

*Address length must always be 24-bits

Absolute Long Address

D1 = $00000000
MOVE.W $001000,D1

Memory

$1000	$12
$1001	$34
$1002	$FF
$1003	$56
$1004	$AA
$1005	$AC
$1006	$12

D1 =

0	0	0	0	1	2	3	4

Immediate Data

- Used to transfer constant values into registers/memory locations.
- Consists of 2 parts:
  - The constant value.
  - The register/memory location to store it in.
- Symbol '#' must be put in front of the constant value.

Types of Constant

Symbol	Data Type	Example
%	Binary	#%01101010
@	Octal	#@123
<none>	Decimal	#45
$	Hexadecimal	#$35
' '	Character	#'B'

Example: Moving Decimal Value to Data Register

D0 = $FFFFFFFF

MOVE.B #12,D0

- Constant value: 12_D = #$0C

D0 =	F	F	F	F	F	F	F	F
Final D0 =	F	F	F	F	F	F	0	C

Example: Moving Hex Value to Memory

MOVE.W #$1234,$004000

- Constant value: $1234 (hex)
- Target: memory address $4000.

Memory Address:

$3FFE	…
$3FFF	…
$4000	$12
$4001	$34
$4002	…

Example: Moving Hex Value to Address Register

A3 = $00000000

MOVEA.L #$00400400,A3

- Constant variable: 00400400 (hex)
- Target: A3.

A3 = $00000000

A3 = $00400400

Example: Moving Hex Value to Memory

MOVE.W #$1234,$004000

- Constant value: $1234 (hex)
- Target: memory address $4000.

Memory Address:

$3FFE	…
$3FFF	…
$4000	$12
$4001	$34
$4002	…

Example: Moving Character Value to Memory

MOVE.L #'BUKU',D0

- 'B' = $42, 'U' = $55, 'K' = $4B, 'U' = $55
- Target: D0.

D0 =

4	2	5	5	4	B	5	5
'B'		'U'		'K'		'U'	

Example: Moving Binary Value to Memory

MOVE.B #%10101011,D0

- %1010 = $A, %1011 = $B
- Target: D0.

D0 =

0	0	0	0	0	0	A	B

Quick Immediate Data

- Similar to ID, but can only transfer 1 byte.
- Byte is sign-extended to 32-bits.
- Must be used together with MOVEQ instruction.
- Can only be used for D_n.

Example: Quick Immediate Data

D1 = $00000000

MOVEQ #$05,D1

- Constant variable: 05 (hex)
- Target: D1.

D1 = $00000000

D1 = $00000005

$05 = 0000 0101

(MSB is 0)

D1 = $00000005

Sign-extended to 32-bits

Example: Quick Immediate Data

D0 = $00000000

MOVEQ #$EA,D0

- Constant value: EA (hex)
- Target: D0.

D0 = $000000EA

D0 = $000000EA

$EA = 1110 1010

(MSB is 1)

D0 = $FFFFFFEA

Sign-extended to 32-bits

Example: Quick Immediate Data

D1 = $FFFFFFFF

MOVEQ #$05,D1

- Constant value: 05 (hex)
- Target: D1.

D1 = $FFFFFFFF

D1 = $FFFFFF05

$05 = 0000 0101

(MSB is 0)

D1 = $00000005

Sign-extended to 32-bits

Implied Addressing

- Uses mnemonics to refer to M68k's internal registers.
- Examples:
  - SR – Status Register
  - USP – User Stack Pointer.
  - SSP – Supervisor Stack Pointer.
  - CCR – Condition Code Register
  - TRAPV – Trap exception if V-bit set.

IA Example – Set Trace Mode

- ORI.W #$8000,SR

	T		S			I_2	I_1	I_0				X	N	Z	V	C
OLD SR=	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0
OR	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NEW SR=	1	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0

Sets trace mode to on (T = 1), other bits unchanged.

IA Example – Clear CCR

- ANDI.B #$00,CCR

				X	N	Z	V	C
OLD CCR =				1	1	0	1	0
AND	0	0	0	0	0	0	0	0
NEW CCR =	0	0	0	0	0	0	0	0

Clears all bits in CCR

Conclusion

We have covered these addressing modes

Dn and An

DRD	D0 → D7
ARD	A0 → A7

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(An,Dn/An.s)
ALA	$001001
ASA	$FFAA
IA	CCR, SR

Immediate data: <id>

ID #($ % @ ‘’)

Conclusion

- Addressing modes allow flexibility in accessing registers memory locations.
- Try to understand how to calculate EA:
  - ARI (PI, PD, D, I)
- The rest are straightforward.

The End

Please read:

Antonakos, pg. 47-58