Addressing Modes

DEALING WITH OPERANDS

Addressing Modes and Operands

- Operands of an instruction can be a register, memory, or immediate value however how we specify these (especially memory values) can be done with a variety of methods which we call *addressing modes*
- Addressing modes refer to the methods an instruction can use to specify the location of an operand
  - RISC approach: few, simple addressing modes
  - CISC approach: complex, multi-operation modes
- Definition: EA = effective address
  - The final *location* of the operand the instruction will use

Shorthand Notation

- M[x] = Value in memory @ address x
- D[n] or A[n] = Register value of Dn or An
- R[n] = Either data or address register value
- Examples:
  - M[4] = D0
    - Memory at address 4 gets the value of D0
  - M[A[1]] = D1
    - Memory at the address specified by A1 gets the value of D1

Overview

Location of Operand	Addressing Modes	Operand	Assembly Notation
Register Contents	**Data Register Direct**	**D[n]**	**D0**
	Address Register Direct	**A[n]**	**A1**
Memory Location	**Address Register Indirect (A.R.I.)**	**M[A[n]]**	**(A2)**
	A.R.I. w/ postincrement	**Operand: M[A[n]] Side Effect: A[n] = A[n] + {1,2,4}**	**(A2)+**
	A.R.I. w/ predecrement	**Operand: M[A[n] – {1,2,4}] Side Effect: A[n] = A[n] – {1,2,4}**	**-(A2)**
	A.R.I. w/ displacement	**M[A[n] + displacement]**	**(0x24,A2)**
	A.R.I. w/ scaled index & displacement	**M[A[n] + disp. + R[m]*size]**	**(0x24,A2,D0.L*4)**
	Absolute Addressing Short	**M[address] (e.g. M[0x7060])**	**0x7060**
	Absolute Addressing Long	**M[address] (e.g. M[0x18000])**	**0x18000**
Immediate	**Immediate**	**Immediate**	**#0x7800**

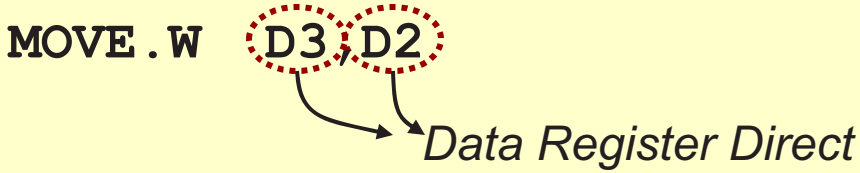
Register Operands

- Two different modes for different register types
  - Data Registers
  - Address Registers

Data Register Direct

- Specifies the contents of a data register

Example:



Data Register Direct

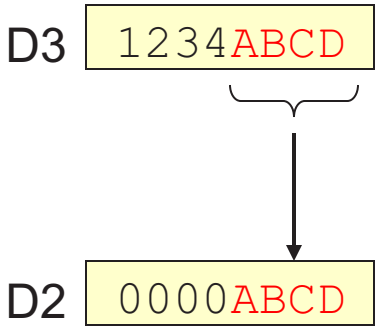
Example:

① *Initial Conditions:* D[3] = 1234ABCD, D[2] = 00000000

```
MOVE.W   D3,D2
```

② Lower Word of D3 put into D2

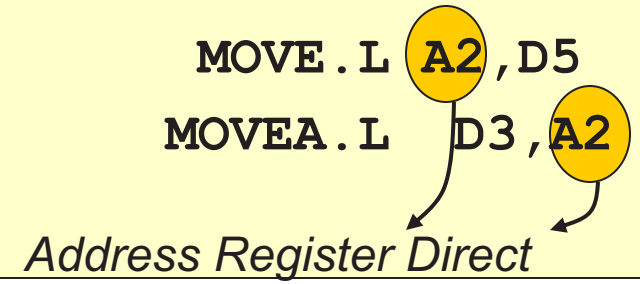
③ D3 retains same value



Address Register Direct

- Specifies the contents of a address register
- Use MOVEA, ADDA, SUBA when address register is destination (though not when source)
- Recommendation: ALWAYS use size .L when destination is address register
  - If you use size .W and destination is an address register, the result will be sign-extended to fill the entire register anyways

Example:



Address Register Direct

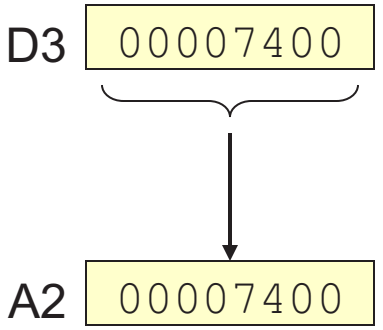
Example:

① *Initial Conditions:* D[3] = 00007400, A[2] = 000F0420

```
MOVEA.L   D3,A2
```

② Longword of D3 put into A2

③ D3 retains same value



Address Register Direct

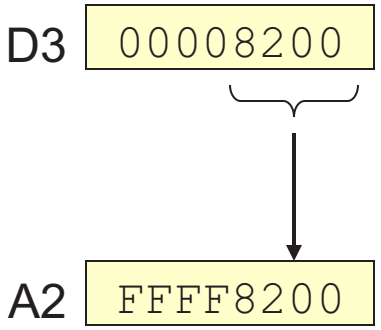
Example:

① *Initial Conditions:* D[3] = 00008200, A[2] = 00007000

```
MOVEA.W   D3,A2
```

② Word of D3 put into A2 and then sign extended to fill all 32 bits

③ D3 retains same value



Notice the sign extenstion – MSB of 8200 = '1'…that '1' is extended to fill all the upper bits…so your address is now 0xFFFF8200

Memory Operands

- 5 Modes for using an address register to specify the address of the memory location
  - Address Register Indirect
  - Address Register Indirect w/ Postincrement
  - Address Register Indirect w/ Predecrement
  - Address Register Indirect w/ Displacement
  - Address Register Indirect w/ Index
- 2 Modes for specifying an address as part of the instruction (called an absolute address)
  - Absolute Short Address Mode
  - Absolute Long Address Mode

Address Register Indirect

- Specifies a memory location
- Use contents of A*n* as an address (pointer) to the actual data
- Similar idea as pointers in *C/C++*
  - (A0) in assembly ⇔ *ptr_A0 in C/C++

Example:

```
MOVE.L (A0),D2
```

Addr. Reg. Indirect

Address Register Indirect

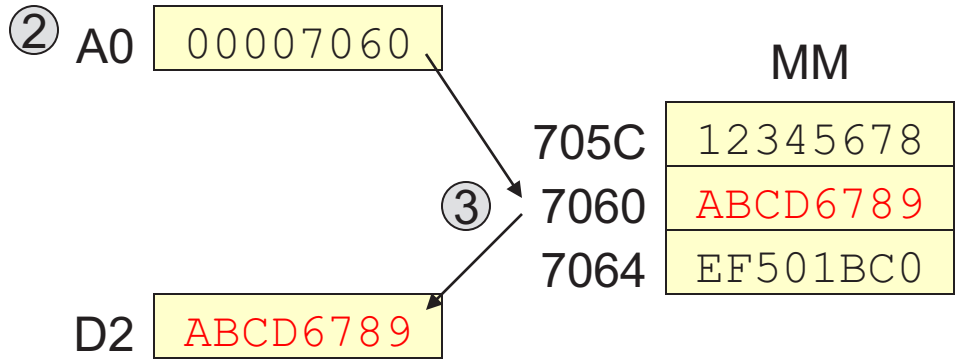
Example:

① *Initial Conditions:* A[0] = 00007060

```
MOVE.L   (A0),D2
```

② <EA> = contents of A0

③ Use <EA> to access memory

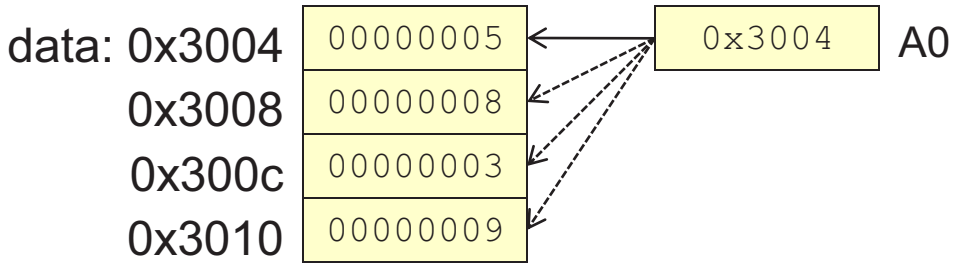


Address Register Indirect

- Good for use as a pointer

```
int data[4]={5,8,3,9};

for(i=0; i<4; i++)
  data[i] = 1;
```

C Code



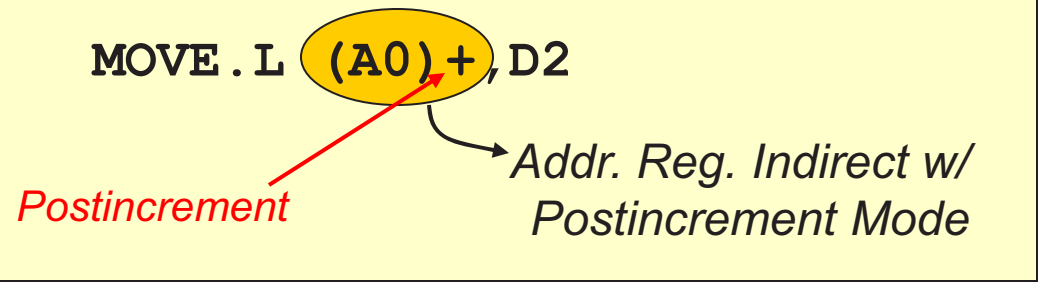
MM

```
movea.l #0x3004,a0
loop 4 times {

  move.l #1,(a0)
  adda.l  #4,a0

}
```

Address Register Indirect w/ Postincrement

- Specifies a memory location
- Use contents of A*n* as address to actual data
- After accessing data, contents of An are incremented by 1, 2, or 4 depending on size (.B, .W, .L)
- Similar idea as pointers in *C/C++*
  - (A0)+ in assembly ⇔ *ptr_A0; ptr_A0++; in C/C++

Example:



Address Register Indirect w/ Postincrement

Example:

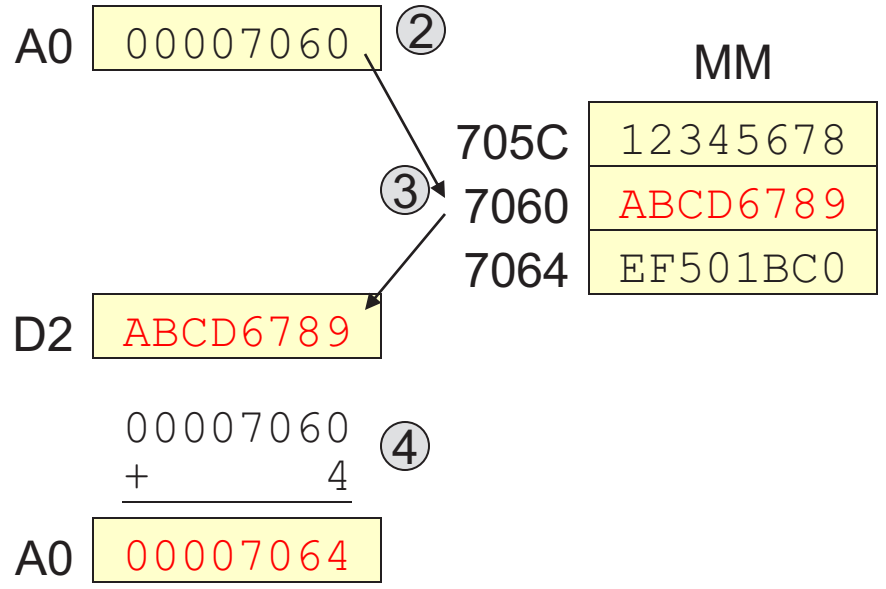
① *Initial Conditions:* A[0] = 00007060

```
MOVE.L   (A0)+,D2
```

② <EA> = contents of A0

③ Increment An by 1, 2, 4 depending on size (.B, .W, .L)

④ Use <EA> to access memory



Address Register Indirect w/ Postincrement

Another Example:

① *Initial Conditions:* A[0] = 00007060

```
MOVE.W   (A0)+,(A0)+
```

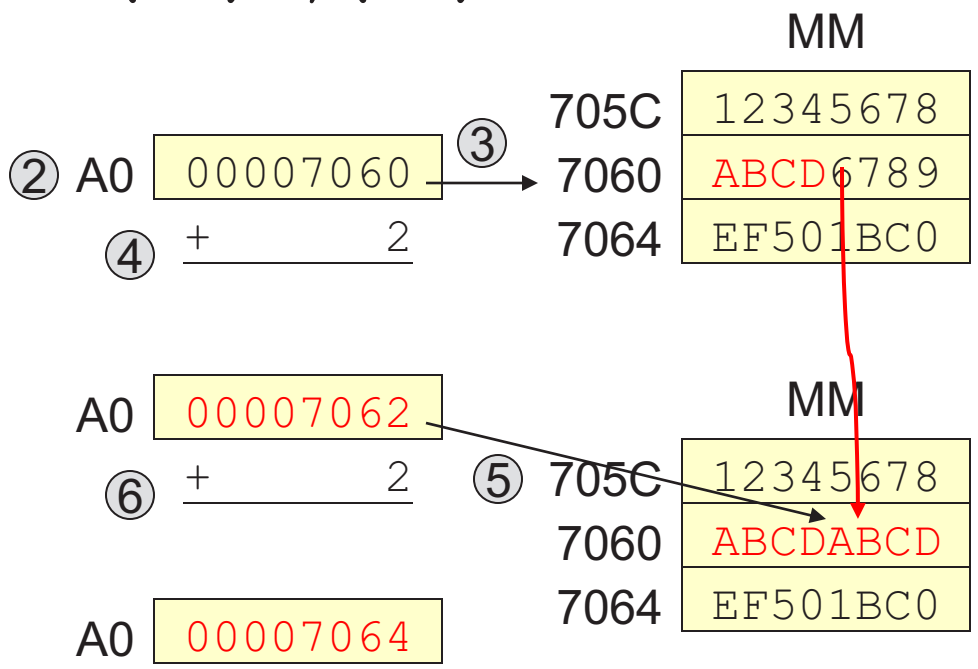
② Source <EA> = contents of A0

③ Use <EA> to access memory

④ Increment A0 by 2 because it is a .W instruction

⑤ Use new contents of A0 as address for destination

⑥ Increment A0 by 2 again



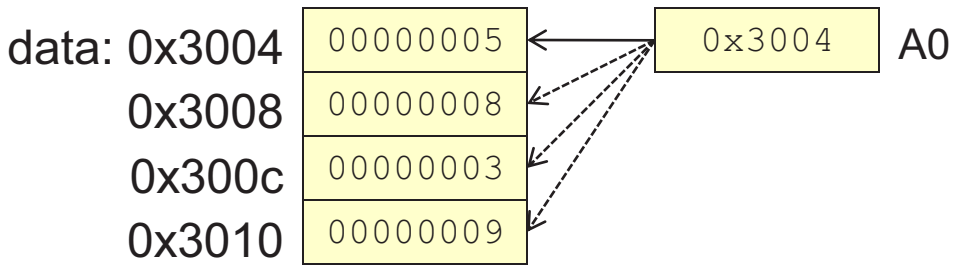
Summary: Work from left to right

Address Register Indirect w/ Postincrement

- Good for use as a pointer moving through an array

```
int data[4]={5,8,3,9};

for(i=0; i<4; i++)
  data[i] = 1;
```

C Code



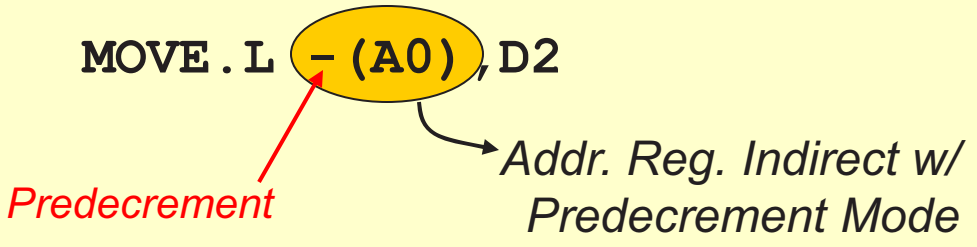
MM

```
movea.l #0x3004,a0
loop 4 times {

  move.l #1,(a0)+

}
```

Address Register Indirect w/ Predecrement

- Specifies a memory location
- Use contents of A*n* as address to actual data
- **Before** accessing data, contents of An are decremented by 1, 2, or 4 depending on the size (.B, .W, .L)
- Similar idea as pointers in *C/C++*
  - `-(A0)` in assembly ⇔ `--ptr_A0; *ptr_A0;` in C/C++

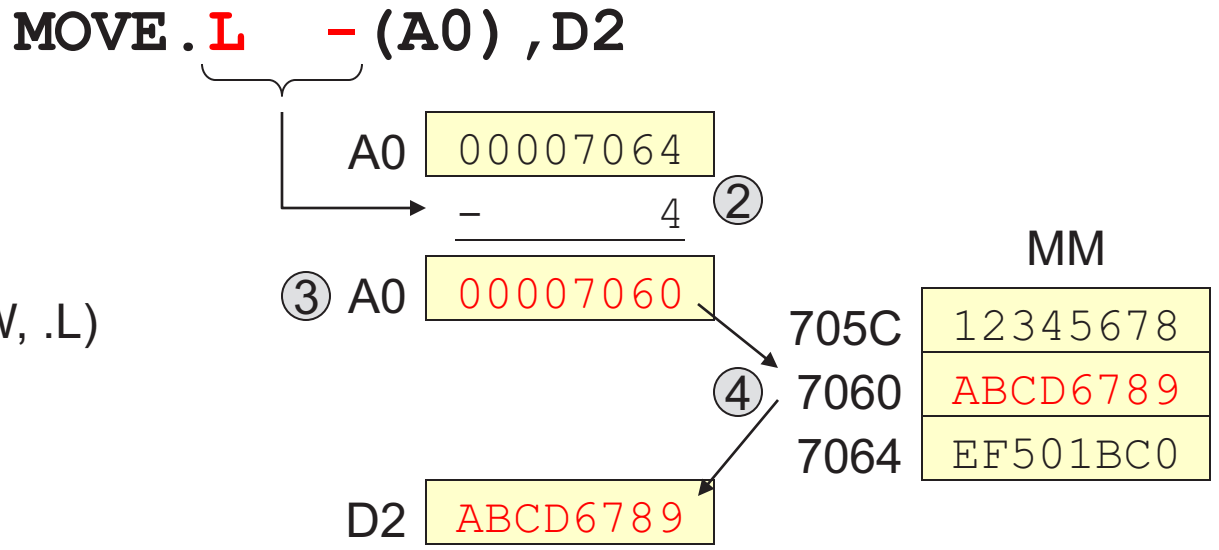
Example:



Address Register Indirect w/ Predecrement

Example:

① *Initial Conditions:* A[0] = 00007064



② Decrement A0 by 1, 2, 4 depending on size (.B, .W, .L)

③ Use new A0 to access memory

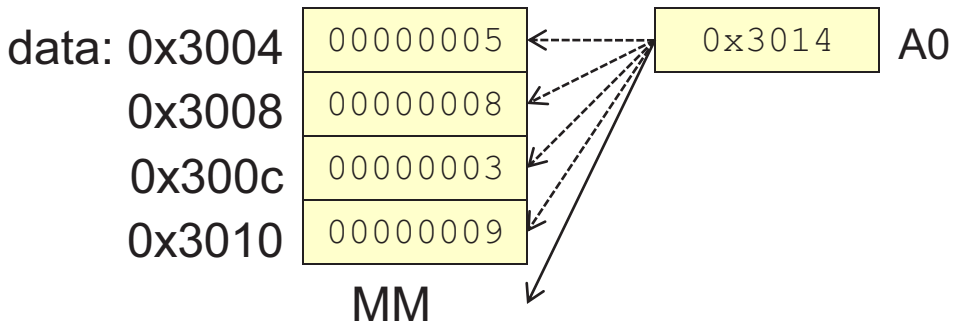
④ <EA> = contents of A0

Address Register Indirect w/ Predecrement

- Good for use as a pointer moving through an array

```
int data[4]={5,8,3,9};

for(i=0; i<4; i++)
  data[i] = 1;
```

C Code




```
movea.l #0x3014,a0
loop 4 times {

  move.l #1,-(a0)

}
```

Address Register Indirect w/ Displacement

- Specifies a memory location
- Use A*n* as base address and adds a displacement value to come up w/ effective address (<EA>)
- Displacement limited to 16-bit signed number
- A*n* not affected (maintains original address)

Example:

```
MOVE.L  (0x24,A0),D2
```

Displacement Value

Addr. Reg. Indirect w/ Displacement Mode

Address Register Indirect w/ Displacement

Example:

① *Initial Conditions:* A[0] = 0000703C

```
MOVE.L   (0x24,A0),D2
```

② <EA> = A[0] + Displacement

③ Use <EA> to access memory

④ <EA> discarded (A0 is left w/ original value)

```
703C = (A0)
+ 0024 = Disp.
7060 = <EA>
```

MM

705C	12345678
7060	ABCD6789
7064	EF501BC0

D2	ABCD6789
A0	0000703C

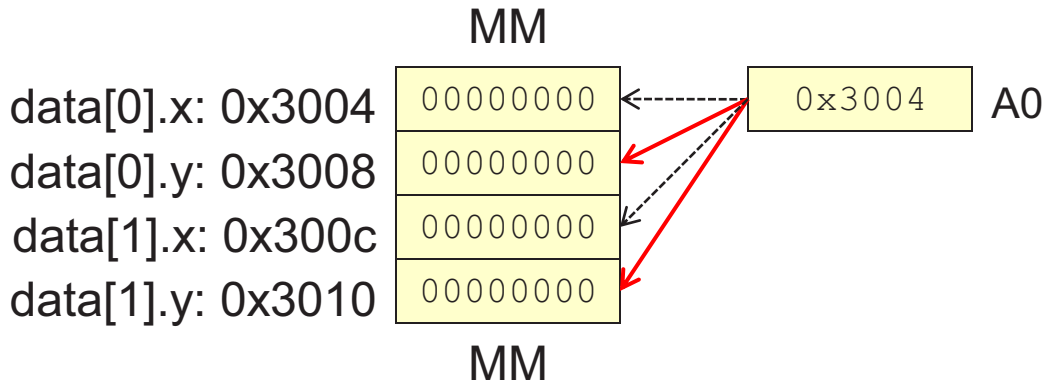
A0 is left unchanged

Address Register Indirect w/ Displacement

- Good for use as a pointer to access fields of a record/structure/class

```
struct mystruct {
  int x;
  int y;

} data[2];

for(i=0; i<2; i++)
  data[i].y = 1;
```

C Code




```
movea.l #0x3004,a0
move.l  #1,d0
loop 2 times {

  move.l  d0,(4,a0)
  adda.l  #8,a0

}
```

Address Register Indirect w/ Index

- Specifies a memory location
- Use A*n* as base address and adds a displacement value and product of the contents of another Data or Address register times a scale factor (1,2,4) to come up w/ effective address (<EA>)
- Displacement limited to 8-bit signed number
- A*n* not affected (maintains original address)

Example:

```
MOVE.L  (0x24,A0,D1.L*4),D2
```

Displacement Value

Addr. Reg. Indirect w/ Index Mode

Address Register Indirect w/ Index

Example:

① *Initial Conditions:* A[0] = 0000782C, D[1] = 00000004

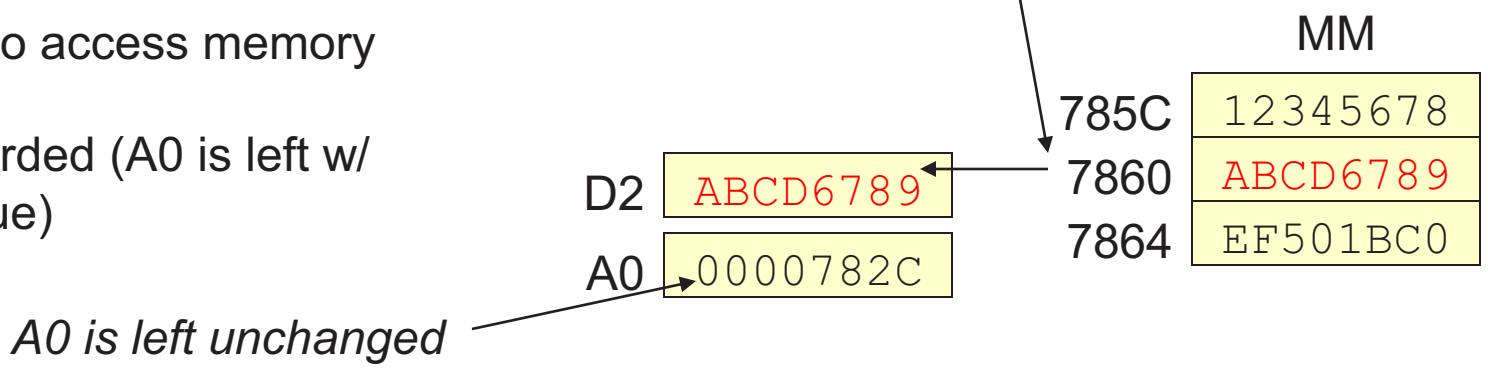
```
MOVE.L   (0x24,A0,D1.L*4),D2
```

```
  0000782C  = (A0)
  00000010  = (D1)*4
+ 00000024  = Disp.
  00007860  = <EA>
```

② <EA> = A[0] + D[1] + Displacement

③ Use <EA> to access memory

④ <EA> discarded (A0 is left w/ original value)



Address Register Indirect w/ Index

Example:

① *Initial Conditions:* A[0] = 0000703C, D[1] = FFFFFF00

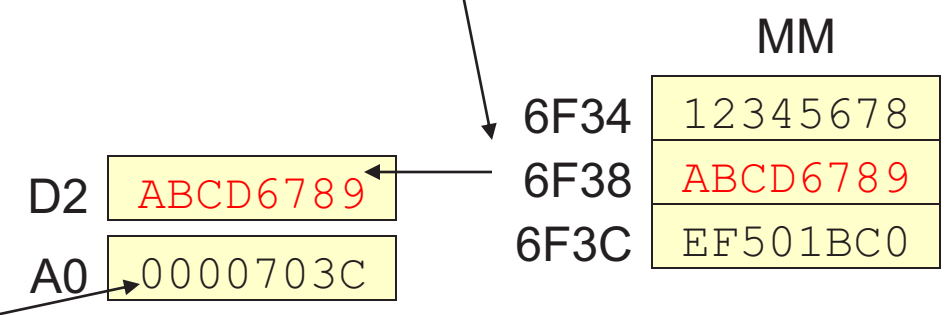
```
MOVE.L   (0xFC,A0,D1.L*4),D2
```

```
  0000703C  = (A0)
  FFFFFF00  = (D1)
+ FFFFFFFC  = Disp.
  00006F38  = <EA>
```

② <EA> = A[0] + D[1] + Displacement

③ Use <EA> to access memory

④ <EA> discarded (A0 is left w/ original value)



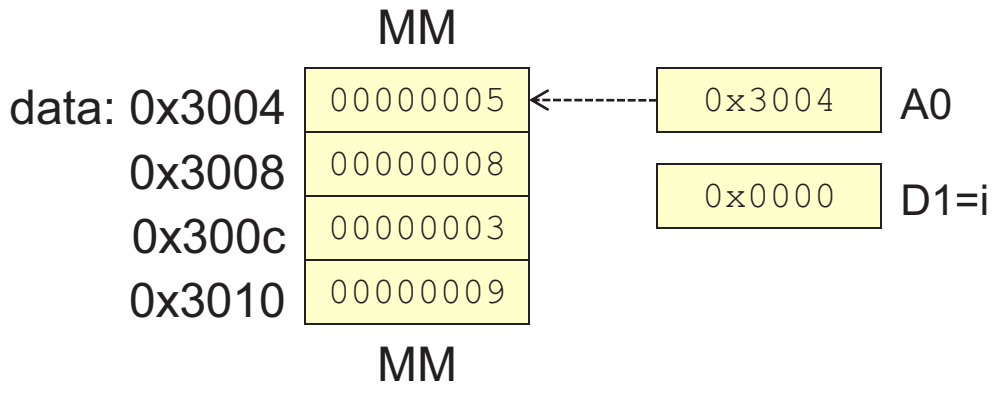
A0 is left unchanged

Address Register Indirect w/ Index

- Good for use as a pointer to access an array

```
int data[4] = {5,8,3,9);

for(i=0; i<4; i++)
  data[i] = 1;
```

C Code



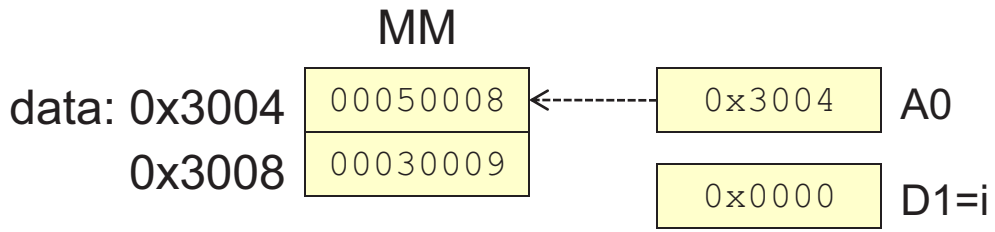

```
movea.l #0x3004,a0 ; base ptr.
move.l #1,d0       ; val to move
move.l #0,d1       ; i cntr.
loop 4 times {

  move.l d0,(0,a0,d1.L*4)
  addi.l #1,d1

}
```

Address Register Indirect w/ Index

- Good for use as a pointer to access an array

```
short data[4] = {5,8,3,9);

for(i=0; i<4; i++)
  data[i] = 1;
```

C Code



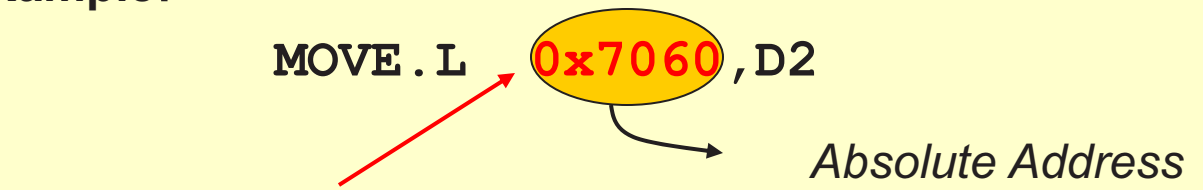
MM

```
movea.l #0x3004,a0 ; base ptr.
move.l #1,d0       ; val to
movemove.l #0,d1       ; i cntr.
loop 4 times {

  move.l d0,(0,a0,d1.L*2)
  addi.l #1,d1

}
```

Short Absolute Address

- Specifies the exact memory location
- Short address requires only 16-bits but MSB must be 0 (∴ 0000-7FFF)

Example:



Short Absolute Address

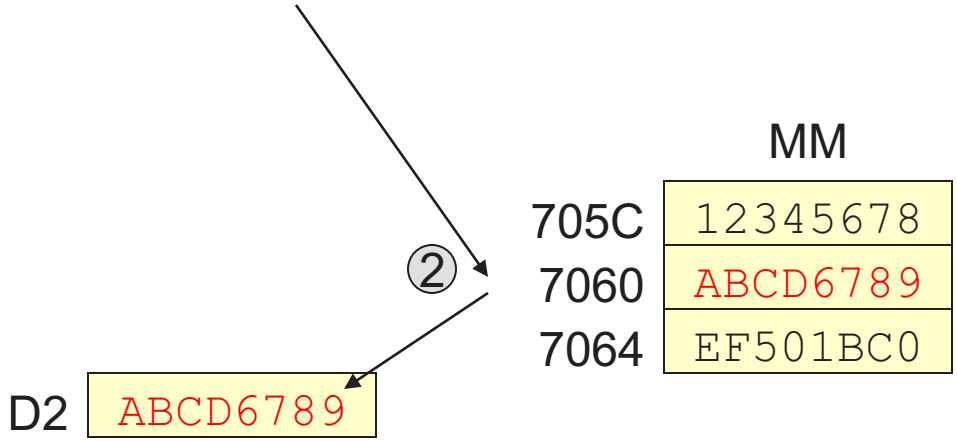
Example:

This is considered a short address because it is <= 0x7FFF

```
MOVE.L   0x7060,D2
```

① <EA> = 0x7060

② Access Longword at 0x7060



Long Absolute Address

- Specifies the exact memory location
- Long address requires more than 16-bits (also includes when Short address has MSB=1 ∴ 8000-FFFFFF)

Example:

```
MOVE.L  0x18060,D2
```

Long Absolute Address

Absolute Address Mode

Long Absolute Address

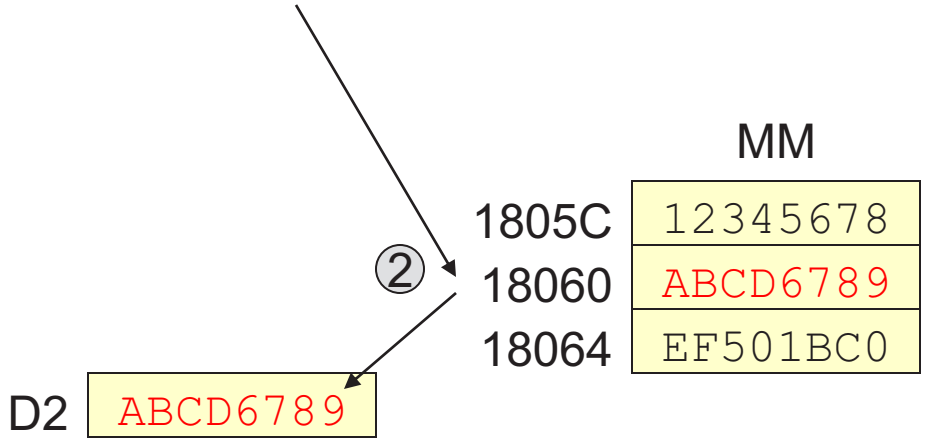
Example:

This is considered a long address because it is >= 0x8000

```
MOVE.L   0x18060,D2
```

① <EA> = 0x18060

② Access Longword at 0x18060



Immediate Operands

- One mode indicating the operand is stored as part of the instruction
  - Immediate Mode

Immediate Mode

- Places the exact data into the destination
- '#' indicates Immediate Mode
- The immediate value will be zero-extended to the size indicated by the instruction

Example:

```
MOVE.L  #0x7060,D2
```

Immediate Value

Immediate Mode

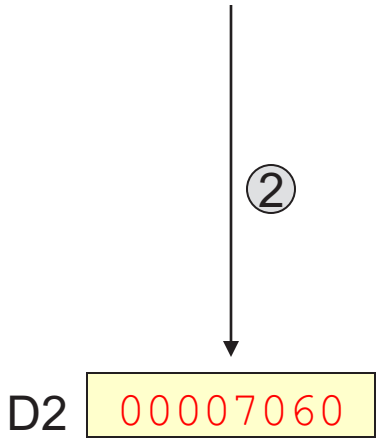
Immediate Mode

Example:

① Initial Conditions: (D2) = ABCD1234

```
MOVE.L   #0x7060,D2
```

② Move constant 0x00007060 to D2



Overview

Location of Operand	Addressing Modes	Operand	Assembly Notation
Register Contents	**Data Register Direct**	**D[n]**	**D0**
	Address Register Direct	**A[n]**	**A1**
Memory Location	**Address Register Indirect (A.R.I.)**	**M[A[n]]**	**(A2)**
	A.R.I. w/ postincrement	**M[A[n]]**	**(A2)+**
	A.R.I. w/ predecrement	**M[A[n] – {1,2,4}]**	**-(A2)**
	A.R.I. w/ displacement	**M[A[n] + displacement]**	**(0x24,A2)**
	A.R.I. w/ scaled index & displacement	**M[A[n] + disp. + R[m]*size]**	**(0x24,A2,D0.L*4)**
	Absolute Addressing Short	**M[address] (e.g. M[0x7060])**	**0x7060**
	Absolute Addressing Long	**M[address] (e.g. M[0x18000])**	**0x18000**
Immediate	**Immediate**	**Immediate**	**#0x7800**

Addressing Mode Examples

Initial Contents: A[0] = 0, A[1] = 0, D[0] = 3

		(A0)	(A1)
1	MOVEA.L #0x00007000,A0	0x00007000	
2	MOVEA.L #0x00007008,A1		
3	MOVE.B (A0)+,(A1)+		
4	MOVE.B (A0)+,0x7(A0)		
5	ADDA.L #1,A0		
6	MOVE.B (A0),1(A1)		
7	MOVE.B -(A0),-1(A1,D0.L)		

MOVEA.L #0x00007000,A0

Main Memory

7000	1A 1B 1C 1D
7004	00 00 00 00
7008	00 00 00 00

Addressing Mode Examples

Initial Contents: A[0] = 0, A[1] = 0, D[0] = 3

		(A0)	(A1)
1	MOVEA.L #0x00007000,A0	0x00007000	
2	MOVEA.L #0x00007008,A1		0x00007008
3	MOVE.B (A0)+,(A1)+		
4	MOVE.B (A0)+,0x7(A0)		
5	ADDA.L #1,A0		
6	MOVE.B (A0),1(A1)		
7	MOVE.B -(A0),-1(A1,D0.L)		

MOVEA.L #0x00007008,A1

Main Memory

7000	1A 1B 1C 1D
7004	00 00 00 00
7008	00 00 00 00

Addressing Mode Examples

Initial Contents: A[0] = 0, A[1] = 0, D[0] = 3

		(A0)	(A1)
1	MOVEA.L #0x00007000,A0	0x00007000	
2	MOVEA.L #0x00007008,A1		0x00007008
3	MOVE.B (A0)+,(A1)+	0x00007001	0x00007009
4	MOVE.B (A0)+,0x7(A0)		
5	ADDA.L #1,A0		
6	MOVE.B (A0),1(A1)		
7	MOVE.B -(A0),-1(A1,D0.L)		

MOVE.B (A0)+,(A1)+

1. **Get Value pointed to by A0 => 0x1A**
2. **Increment A0 => 0x7001**
3. **Place value in location pointed to by A1 => 0x7008**
4. **Increment A1 => 0x7009**

Main Memory

7000	1A	1B	1C	1D
7004	00	00	00	00
7008	1A	00	00	00

Addressing Mode Examples

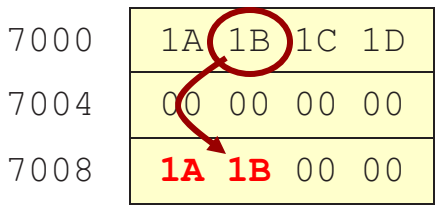
Initial Contents: A[0] = 0, A[1] = 0, D[0] = 3

		(A0)	(A1)
1	MOVEA.L #0x00007000,A0	0x00007000	
2	MOVEA.L #0x00007008,A1		0x00007008
3	MOVE.B (A0)+,(A1)+	0x00007001	0x00007009
4	MOVE.B (A0)+,0x7(A0)	0x00007002	
5	ADDA.L #1,A0		
6	MOVE.B (A0),1(A1)		
7	MOVE.B -(A0),-1(A1,D0.L)		

MOVE.B (A0)+, 0x7(A0)

1. **Get Value pointed to by A0 => 0x1B**
2. **Increment A0 => 0x7002**
3. **Place value in location pointed to by 7+(A0) => 0x7009**

Main Memory



Addressing Mode Examples

Initial Contents: A[0] = 0, A[1] = 0, D[0] = 3

		(A0)	(A1)
1	MOVEA.L #0x00007000,A0	0x00007000	
2	MOVEA.L #0x00007008,A1		0x00007008
3	MOVE.B (A0)+,(A1)+	0x00007001	0x00007009
4	MOVE.B (A0)+,0x7(A0)	0x00007002	
5	ADDA.L #1,A0	0x00007003	
6	MOVE.B (A0),1(A1)		
7	MOVE.B -(A0),-1(A1,D0.L)		

ADDA.L #1,A0

1. Add 1 to (A0) => 0x7003

Main Memory

7000	1A 1B 1C 1D
7004	00 00 00 00
7008	**1A 1B** 00 00

Addressing Mode Examples

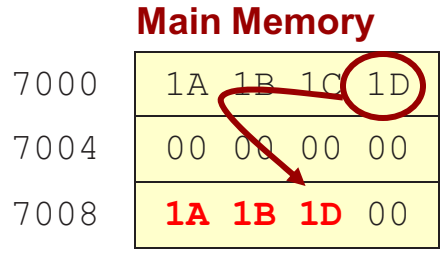
Initial Contents: A[0] = 0, A[1] = 0, D[0] = 3

		(A0)	(A1)
1	MOVEA.L #0x00007000,A0	0x00007000	
2	MOVEA.L #0x00007008,A1		0x00007008
3	MOVE.B (A0)+,(A1)+	0x00007001	0x00007009
4	MOVE.B (A0)+,0x7(A0)	0x00007002	
5	ADDA.L #1,A0	0x00007003	
6	MOVE.B (A0),1(A1)	0x00007003	0x00007009
7	MOVE.B -(A0),-1(A1,D0.L)		

MOVE.B (A0), 1(A1)

1. **Get Value pointed to by A0 => 0x1D**
2. **Place value in location pointed to by 1+(A1) => 0x700A**

Main Memory



Addressing Mode Examples

Initial Contents: A[0] = 0, A[1] = 0, D[0] = 3

		(A0)	(A1)
1	MOVEA.L #0x00007000,A0	0x00007000	
2	MOVEA.L #0x00007008,A1		0x00007008
3	MOVE.B (A0)+,(A1)+	0x00007001	0x00007009
4	MOVE.B (A0)+,0x7(A0)	0x00007002	
5	ADDA.L #1,A0	0x00007003	
6	MOVE.B (A0),(1,A1)	0x00007003	0x00007009
7	MOVE.B -(A0),(-1,A1,D0.L*1)	0x00007002	0x00007009

MOVE.B -(A0), (-1,A1,D0.L*1)

1. **Decrement A0 by 1 => 0x7002**
2. **Get Value pointed to by A0 => 0x1C**
3. **Place value in location pointed to by -1+(A1)+(D0) => 0x700B**

Main Memory