Addressing Mode Examples

EE 357

Code Example 1

```
          .data     0x20003004
STR:      .ascii    'A2g\n'
VAL       .equ      0x1234
DAT:      .space    8
PTR:      .long     DAT+4

          .text
MAIN:     MOVEA.L   PTR,A0
          MOVE.W    #VAL,(A0)+
          MOVE.W    STR+2,(A0)
          MOVE.L    A0,-6(A0)
          MOVEA.L   #DAT,A1
          MOVEA.L   (A1),A2
          MOVE.W    (A2),D0
          STOP      #0x2700
```

Initial

A0=	00000000
A1=	00000000
A2=	00000000
D0=	00000000

	0x20003004
	0x20003008
	0x2000300c
	0x20003010

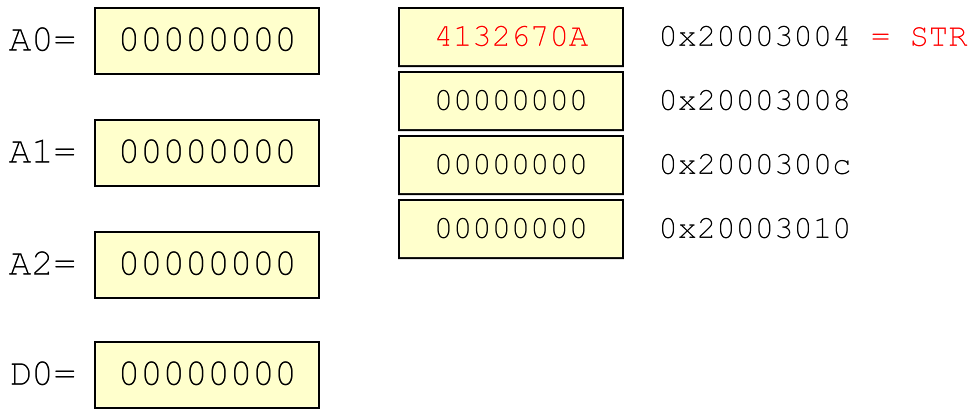
Final

A0=	
A1=	
A2=	
D0=	

	0x20003004
	0x20003008
	0x2000300c
	0x20003010

Code Example 1

```
         .data     0x20003004
STR:     .ascii    'A2g\n'
VAL      .equ      0x1234
DAT:     .space    8
PTR:     .long     DAT+4

         .text
MAIN:    MOVEA.L   PTR,A0
         MOVE.W    #VAL,(A0)+
         MOVE.W    STR+2,(A0)
         MOVE.L    A0,-6(A0)
         MOVEA.L   #DAT,A1
         MOVEA.L   (A1),A2
         MOVE.W    (A2),D0
         STOP      #0x2700
```



- STR evaluates to the address 0x20003004
- Each character in the string 'A2g' is converted to ASCII (0x41, 0x32, 0x67, 0x0a) and stored as separate bytes (\n = ASCII 0x0a)

Code Example 1

```
          .data     0x20003004
STR:      .ascii    'A2g\n'
VAL       .equ      0x1234
DAT:      .space    8
PTR:      .long     DAT+4
          .text
MAIN:     MOVEA.L   PTR,A0
          MOVE.W    #VAL,(A0)+
          MOVE.W    STR+2,(A0)
          MOVE.L    A0,-6(A0)
          MOVEA.L   #DAT,A1
          MOVEA.L   (A1),A2
          MOVE.W    (A2),D0
          STOP      #0x2700
```

A0=	00000000
A1=	00000000
A2=	00000000
D0=	00000000

4132670A	0x20003004 = STR
00000000	0x20003008
00000000	0x2000300c
00000000	0x20003010

- .equ takes up no space in memory; they are translated by the assembler and will replace VAL with 0x1234 anywhere it is used in the code

Code Example 1

```
        .data     0x20003004
STR:    .ascii    'A2g\n'
VAL     .equ      0x1234
DAT:    .space    8
PTR:    .long     DAT+4
        .text
MAIN:   MOVEA.L   PTR,A0
        MOVE.W    #VAL,(A0)+
        MOVE.W    STR+2,(A0)
        MOVE.L    A0,-6(A0)
        MOVEA.L   #DAT,A1
        MOVEA.L   (A1),A2
        MOVE.W    (A2),D0
        STOP      #0x2700
```

A0=	00000000
A1=	00000000
A2=	00000000
D0=	00000000

4132670A	0x20003004 = STR
00000000	0x20003008 = DAT
00000000	0x2000300c
00000000	0x20003010

- The 8 bytes (4 words) starting at 0x20003008 are reserved and left blank for later use in the code

Code Example 1

```
        .data     0x20003004
STR:    .ascii    'A2g\n'
VAL     .equ      0x1234
DAT:    .space    8
PTR:    .long     DAT+4
        .text
MAIN:   MOVEA.L   PTR,A0
        MOVE.W    #VAL,(A0)+
        MOVE.W    STR+2,(A0)
        MOVE.L    A0,-6(A0)
        MOVEA.L   #DAT,A1
        MOVEA.L   (A1),A2
        MOVE.W    (A2),D0
        STOP      #0x2700
```

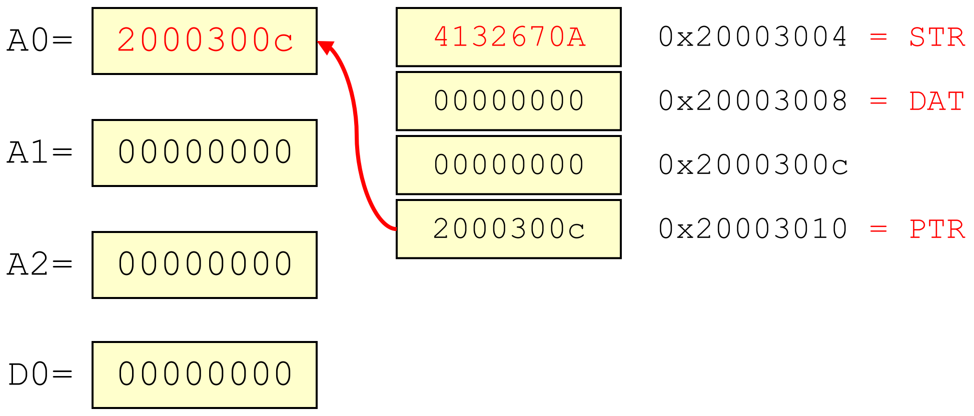
A0=	00000000
A1=	00000000
A2=	00000000
D0=	00000000

4132670A	0x20003004 = STR
00000000	0x20003008 = DAT
00000000	0x2000300c
2000300c	0x20003010 = PTR

- PTR evaluates to the current location of 0x20003010
- DAT + 4 evaluates to 0x20003008 + 4 = 0x2000300c which is stored as a longword starting at 0x20003010

Code Example 1

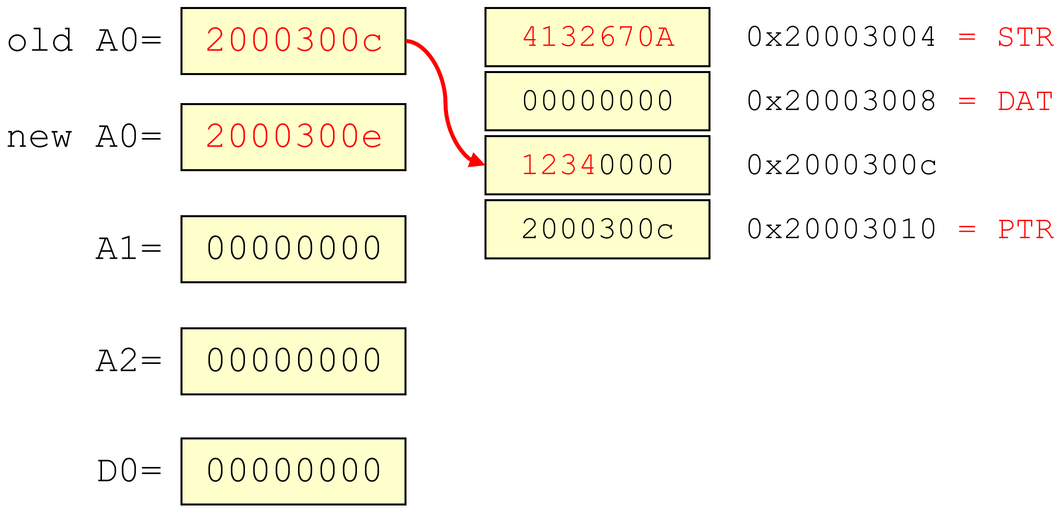
```
            .data     0x20003004
STR:        .ascii    'A2g\n'
VAL         .equ      0x1234
DAT:        .space    8
PTR:        .long     DAT+4

            .text
MAIN:       MOVEA.L   PTR,A0
            MOVE.W    #VAL,(A0)+
            MOVE.W    STR+2,(A0)
            MOVE.L    A0,-6(A0)
            MOVEA.L   #DAT,A1
            MOVEA.L   (A1),A2
            MOVE.W    (A2),D0
            STOP      #0x2700
```



- PTR evaluates to an address of 0x20003010
- This is not an immediate value (denoted with '#'), so we must go get the longword at 0x20003010 and put it in A0.
- 0x2000300c is put in A0

Code Example 1

```
        .data     0x20003004
STR:    .ascii    'A2g\n'
VAL     .equ      0x1234
DAT:    .space    8
PTR:    .long     DAT+4

        .text
MAIN:   MOVEA.L   PTR,A0
        MOVE.W    #VAL,(A0)+
        MOVE.W    STR+2,(A0)
        MOVE.L    A0,-6(A0)
        MOVEA.L   #DAT,A1
        MOVEA.L   (A1),A2
        MOVE.W    (A2),D0
        STOP      #0x2700
```



- '#' indicates an immediate value and VAL is replaced by 0x1234
- Thus, 0x1234 is placed in the word pointed to by A0 which is the word at 0x2000300c
- A0 is then incremented by 2 since this is a word size instruction

Code Example 1

```
        .data     0x20003004
STR:    .ascii    'A2g\n'
VAL     .equ      0x1234
DAT:    .space    8
PTR:    .long     DAT+4
        .text
MAIN:   MOVEA.L   PTR,A0
        MOVE.W    #VAL,(A0)+
        MOVE.W    STR+2,(A0)
        MOVE.L    A0,-6(A0)
        MOVEA.L   #DAT,A1
        MOVEA.L   (A1),A2
        MOVE.W    (A2),D0
        STOP      #0x2700
```

A0=	2000300e
A1=	00000000
A2=	00000000
D0=	00000000

4132670A	0x20003004	= STR
00000000	0x20003008	= DAT
1234670A	0x2000300c	
2000300c	0x20003010	= PTR

- STR+2 evaluates to 0x20003004+2=0x20003006
- This is not an immediate value (denoted with ‘#’), so we must go get the word at 0x20003006 and put at 0x2000300e, the location ***pointed to*** by A0, ***NOT A0 ITSELF***
- 0x670A is put in location 0x2000300e

Code Example 1

```
        .data     0x20003004
STR:    .ascii    'A2g\n'
VAL     .equ      0x1234
DAT:    .space    8
PTR:    .long     DAT+4
        .text
MAIN:   MOVEA.L   PTR,A0
        MOVE.W    #VAL,(A0)+
        MOVE.W    STR+2,(A0)
        MOVE.L    A0,-6(A0)
        MOVEA.L   #DAT,A1
        MOVEA.L   (A1),A2
        MOVE.W    (A2),D0
        STOP      #0x2700
```

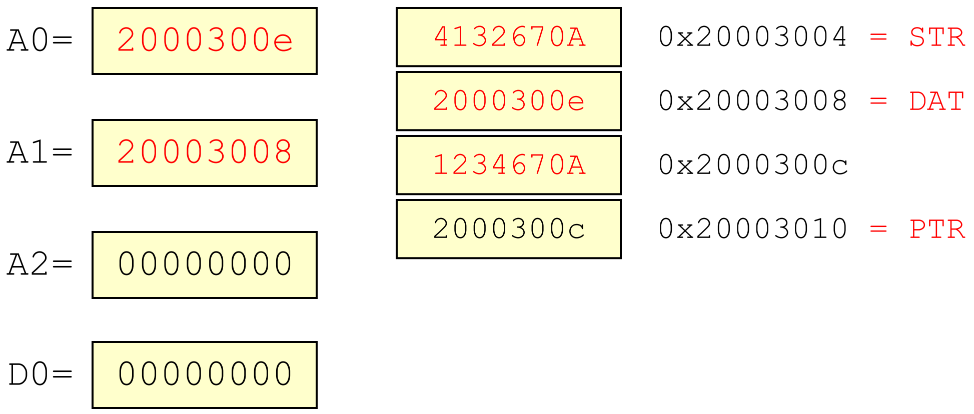
A0=	2000300e
A1=	00000000
A2=	00000000
D0=	00000000

4132670A	0x20003004 = STR
2000300e	0x20003008 = DAT
1234670A	0x2000300c
2000300c	0x20003010 = PTR

- The source operand is A0 w/o parentheses. So we take the contents of A0 = 0x2000300e and put them into the destination location
- The destination operand is the location 0x2000300e – 6 = 0x20003008
- 0x2000300e is put in location 0x20003008 and A0 is left w/ it's original value, 0x2000300e ***(Remember, displacement mode doesn't change the register contents)***

Code Example 1

```
        .data     0x20003004
STR:    .ascii    'A2g\n'
VAL     .equ      0x1234
DAT:    .space    8
PTR:    .long     DAT+4
        .text
MAIN:   MOVEA.L   PTR,A0
        MOVE.W    #VAL,(A0)+
        MOVE.W    STR+2,(A0)
        MOVE.L    A0,-6(A0)
        MOVEA.L   #DAT,A1
        MOVEA.L   (A1),A2
        MOVE.W    (A2),D0
        STOP      #0x2700
```



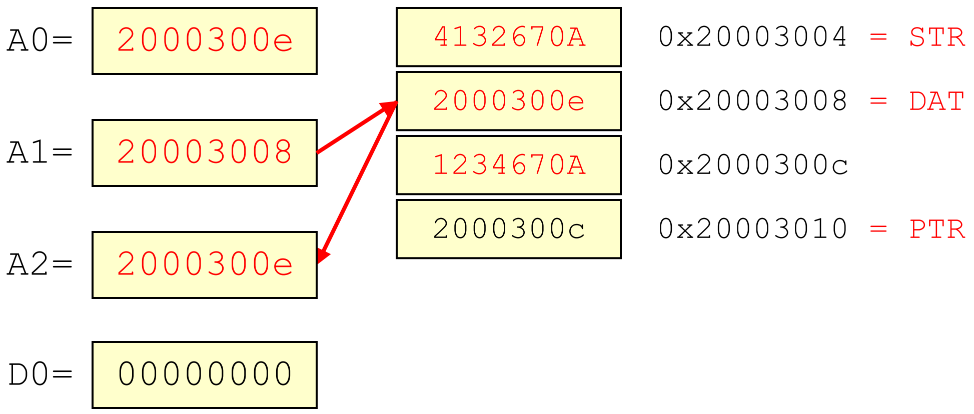
A0= 2000300e
A1= 20003008
A2= 00000000
D0= 00000000
4132670A 0x20003004 = STR
2000300e 0x20003008 = DAT
1234670A 0x2000300c
2000300c 0x20003010 = PTR

- #DAT is an immediate value that evaluates to #0x20003008
- That value, 0x20003008 is put into A1

USC Viterbi School of Engineering

Code Example 1

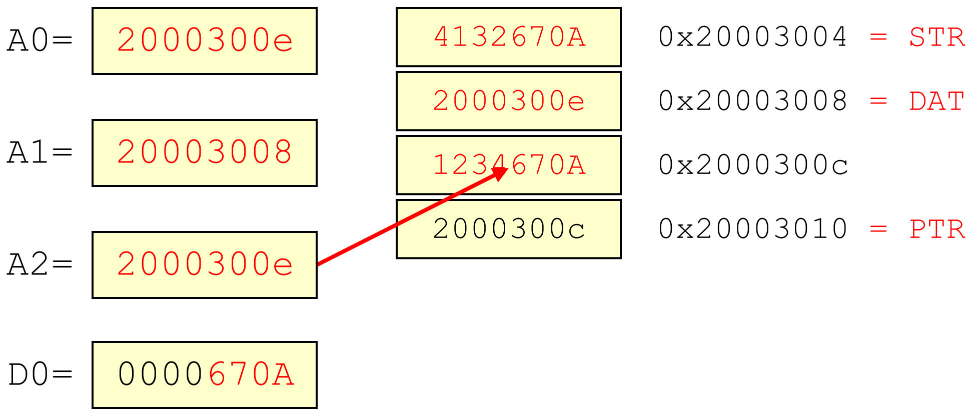
```
        .data      0x20003004
STR:    .ascii     'A2g\n'
VAL     .equ       0x1234
DAT:    .space     8
PTR:    .long      DAT+4
        .text
MAIN:   MOVEA.L    PTR,A0
        MOVE.W     #VAL,(A0)+
        MOVE.W     STR+2,(A0)
        MOVE.L     A0,-6(A0)
        MOVEA.L    #DAT,A1
        MOVEA.L    (A1),A2
        MOVE.W     (A2),D0
        STOP       #0x2700
```



- The source operand is the data pointed to by A1 (it is address register indirect because of the parentheses)
- The word 0x2000300e is placed in A2

Code Example 1

```
            .data      0x20003004
STR:        .ascii     'A2g\n'
VAL         .equ       0x1234
DAT:        .space     8
PTR:        .long      DAT+4
            .text
MAIN:       MOVEA.L    PTR,A0
            MOVE.W     #VAL,(A0)+
            MOVE.W     STR+2,(A0)
            MOVE.L     A0,-6(A0)
            MOVEA.L    #DAT,A1
            MOVEA.L    (A1),A2
            MOVE.W     (A2),D0
            STOP       #0x2700
```



- The source operand is the data pointed to by A2 (it is address register indirect because of the parentheses)
- The word 0x670A is placed in D0

Code Example 1

```
        .data     0x20003004
STR:    .ascii    'A2g\n'
VAL     .equ      0x1234
DAT:    .space    8
PTR:    .long     DAT+4
        .text
MAIN:   MOVEA.L   PTR,A0
        MOVE.W    #VAL,(A0)+
        MOVE.W    STR+2,(A0)
        MOVE.L    A0,-6(A0)
        MOVEA.L   #DAT,A1
        MOVEA.L   (A1),A2
        MOVE.W    (A2),D0
        STOP      #0x2700
```

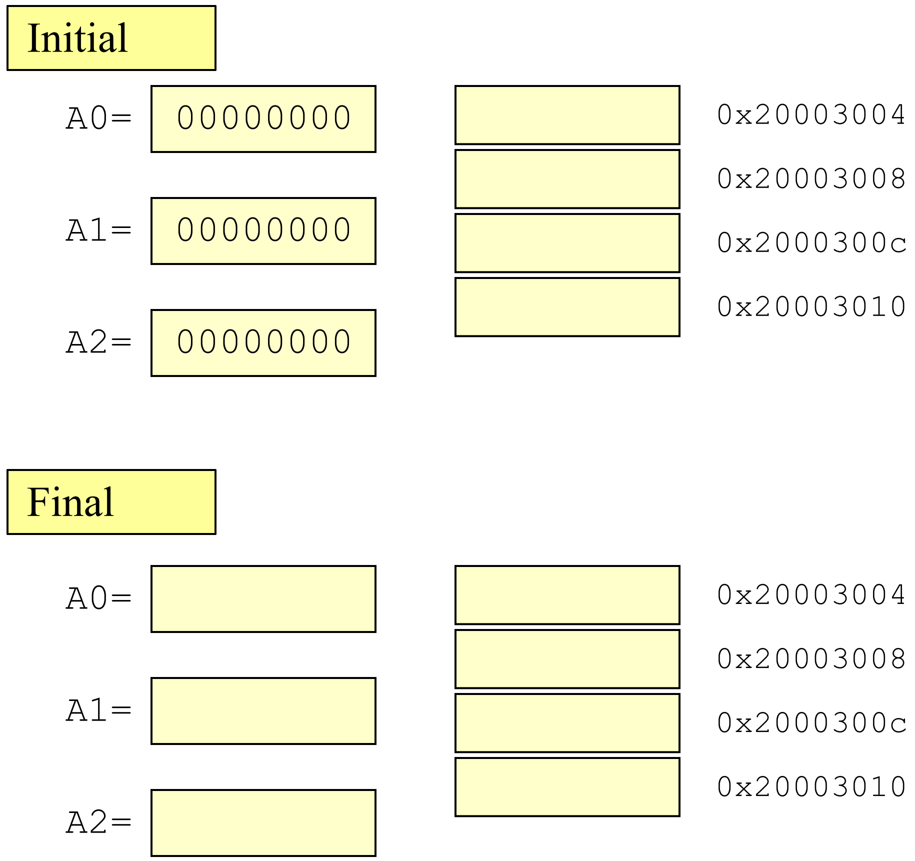
A0=	2000300e
A1=	20003008
A2=	2000300e
D0=	0000670A

4132670A	0x20003004 = STR
2000300e	0x20003008 = DAT
1234670A	0x2000300c
2000300c	0x20003010 = PTR

- After program execution the memory and register have the above contents

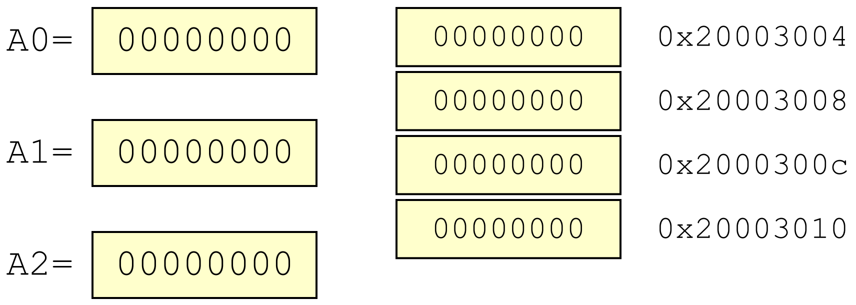
Code Example 2

```
            .data     0x20003004
BUF:        .space    4
CNST        .equ      3
STR:        .asciz    'Hi\n'
PTR:        .long     BUF+4

            .text
MAIN:       MOVEA.L   #PTR,A0
            MOVE.W    STR,-8(A0)
            MOVEA.L   (A0),A1
            MOVEA.L   #BUF+2,A2
            MOVE.B    2(A1),(A2)+
            MOVE.B    CNST(A1),(A2)+
            STOP      #0x2700
```



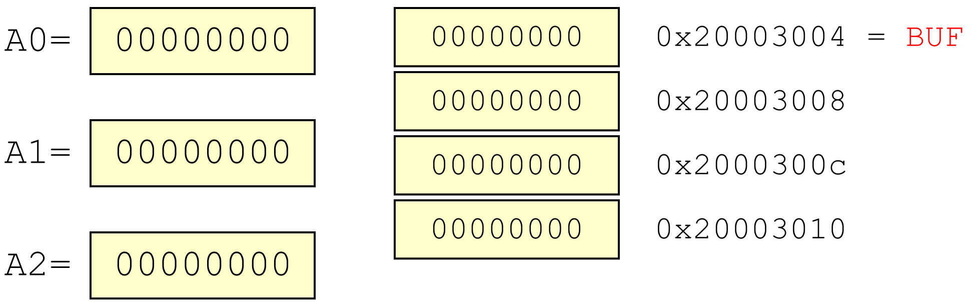
Code Example 2

```
            .data     0x20003004
BUF:        .space    4
CNST        .equ      3
STR:        .asciz    ‘Hi\n’
PTR:        .long     BUF+4
            .text
MAIN:       MOVEA.L   #PTR,A0
            MOVE.W    STR,-8(A0)
            MOVEA.L   (A0),A1
            MOVEA.L   #BUF+2,A2
            MOVE.B    2(A1),(A2)+
            MOVE.B    CNST(A1),(A2)+
            STOP      #0x2700
```



Code Example 2

```
            .data     0x20003004
BUF:        .space    4
CNST        .equ      3
STR:        .asciz    'Hi\n'
PTR:        .long     BUF+4
            .text
MAIN:       MOVEA.L   #PTR,A0
            MOVE.W    STR,-8(A0)
            MOVEA.L   (A0),A1
            MOVEA.L   #BUF+2,A2
            MOVE.B    2(A1),(A2)+
            MOVE.B    CNST(A1),(A2)+
            STOP      #0x2700
```



- BUF evaluates to the address 0x20003004
- .space 4 – reserves 4 bytes for use later in the program

Code Example 2

```
        .data     0x20003004
BUF:    .space    4
CNST    .equ      3
STR:    .asciz    'Hi\n'
PTR:    .long     BUF+4
        .text
MAIN:   MOVEA.L   #PTR,A0
        MOVE.W    STR,-8(A0)
        MOVEA.L   (A0),A1
        MOVEA.L   #BUF+2,A2
        MOVE.B    2(A1),(A2)+
        MOVE.B    CNST(A1),(A2)+
        STOP      #0x2700
```

```
A0= 00000000

A1= 00000000

A2= 00000000
```

00000000	0x20003004 = BUF
00000000	0x20003008
00000000	0x2000300c
00000000	0x20003010

- .equ takes up no space in memory; they are translated by the assembler and will replace CNST with 3 wherever CNST appears later on

Code Example 2

```
        .data       0x20003004
BUF:    .space      4
CNST    .equ        3
STR:    .asciz      'Hi\n'
PTR:    .long       BUF+4
        .text
MAIN:   MOVEA.L     #PTR,A0
        MOVE.W      STR,-8(A0)
        MOVEA.L     (A0),A1
        MOVEA.L     #BUF+2,A2
        MOVE.B      2(A1),(A2)+
        MOVE.B      CNST(A1),(A2)+
        STOP        #0x2700
```

A0=	00000000
A1=	00000000
A2=	00000000

00000000	0x20003004 = BUF
48690a00	0x20003008 = STR
00000000	0x2000300c
00000000	0x20003010

- STR evaluates to 0x20003008
- Each character in the string ‘Hi \n’ is converted to ASCII [0x48, 0x69, 0x0a (\n),0x00 (Null)]

USC Viterbi School of Engineering

Code Example 2

```
            .data       0x20003004
BUF:        .space      4
CNST        .equ        3
STR:        .asciz      'Hi\n'
PTR:        .long       BUF+4
            .text
MAIN:       MOVEA.L     #PTR,A0
            MOVE.W      STR,-8(A0)
            MOVEA.L     (A0),A1
            MOVEA.L     #BUF+2,A2
            MOVE.B      2(A1),(A2)+
            MOVE.B      CNST(A1),(A2)+
            STOP        #0x2700
```

A0=	00000000
A1=	00000000
A2=	00000000

00000000	0x20003004	= BUF
48690a00	0x20003008	= STR
20003008	0x2000300c	= PTR
00000000	0x20003010	

- PTR evaluates to 0x2000300c
- The longword placed at 0x2000300c is BUF (which evaluates to 0x20003004) + 4 = 0x0000x20003008

Code Example 2

```
          .data     0x20003004
BUF:      .space    4
CNST      .equ      3
STR:      .asciz    'Hi\n'
PTR:      .long     BUF+4
          .text
MAIN:     MOVEA.L   #PTR,A0
          MOVE.W    STR,-8(A0)
          MOVEA.L   (A0),A1
          MOVEA.L   #BUF+2,A2
          MOVE.B    2(A1),(A2)+
          MOVE.B    CNST(A1),(A2)+
          STOP      #0x2700
```

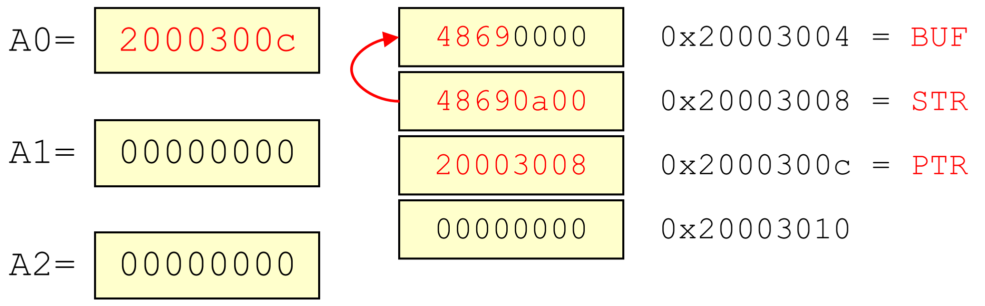
A0=	2000300c
A1=	00000000
A2=	00000000

00000000	0x20003004 = BUF
48690a00	0x20003008 = STR
20003008	0x2000300c = PTR
00000000	0x20003010

- PTR evaluates to an address of 0x2000300c
- This IS an immediate value (denoted with '#'), so we just use the value that PTR evaluates to (i.e. 0x2000300c).
- 0x2000300c is put in A0

Code Example 2

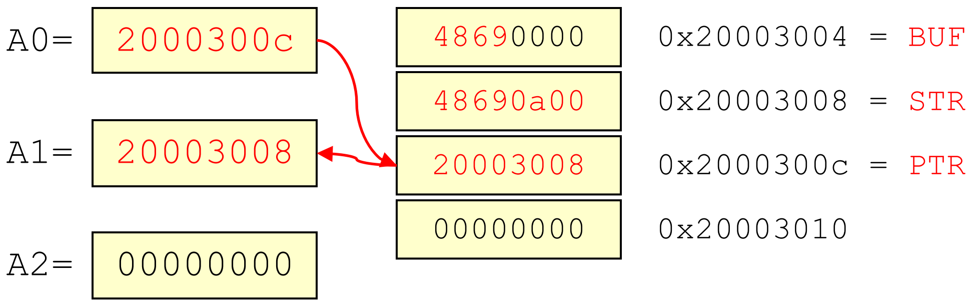
```
        .data     0x20003004
BUF:    .space    4
CNST    .equ      3
STR:    .asciz    'Hi\n'
PTR:    .long     BUF+4

        .text
MAIN:   MOVEA.L   #PTR,A0
        MOVE.W    STR,-8(A0)
        MOVEA.L   (A0),A1
        MOVEA.L   #BUF+2,A2
        MOVE.B    2(A1),(A2)+
        MOVE.B    CNST(A1),(A2)+
        STOP      #0x2700
```


A0= 2000300c
A1= 00000000
A2= 00000000
48690000 0x20003004 = BUF
48690a00 0x20003008 = STR
20003008 0x2000300c = PTR
00000000 0x20003010

- STR evaluates to an address of 0x20003008
- This IS NOT an immediate value (there is no '#'), so we go to that address in memory and get the word there (0x4869).
- We place that word (0x4869) at the address in A0 (0x2000300c) – 8 = 0x20003004

Code Example 2

```
         .data     0x20003004
BUF:     .space    4
CNST     .equ      3
STR:     .asciz    'Hi\n'
PTR:     .long     BUF+4
         .text
MAIN:    MOVEA.L   #PTR,A0
         MOVE.W    STR,-8(A0)
         MOVEA.L   (A0),A1
         MOVEA.L   #BUF+2,A2
         MOVE.B    2(A1),(A2)+
         MOVE.B    CNST(A1),(A2)+
         STOP      #0x2700
```


A0= 2000300c
A1= 20003008
A2= 00000000
48690000 0x20003004 = BUF
48690a00 0x20003008 = STR
20003008 0x2000300c = PTR
00000000 0x20003010

- (A0) tells us to go get the longword at the address in A0 (i.e. go get the data at 0x2000300c) which is 0x20003008
- We then place that longword in A1

Code Example 2

```
        .data     0x20003004
BUF:    .space    4
CNST    .equ      3
STR:    .asciz    'Hi\n'
PTR:    .long     BUF+4

        .text
MAIN:   MOVEA.L   #PTR,A0
        MOVE.W    STR,-8(A0)
        MOVEA.L   (A0),A1
        MOVEA.L   #BUF+2,A2
        MOVE.B    2(A1),(A2)+
        MOVE.B    CNST(A1),(A2)+
        STOP      #0x2700
```

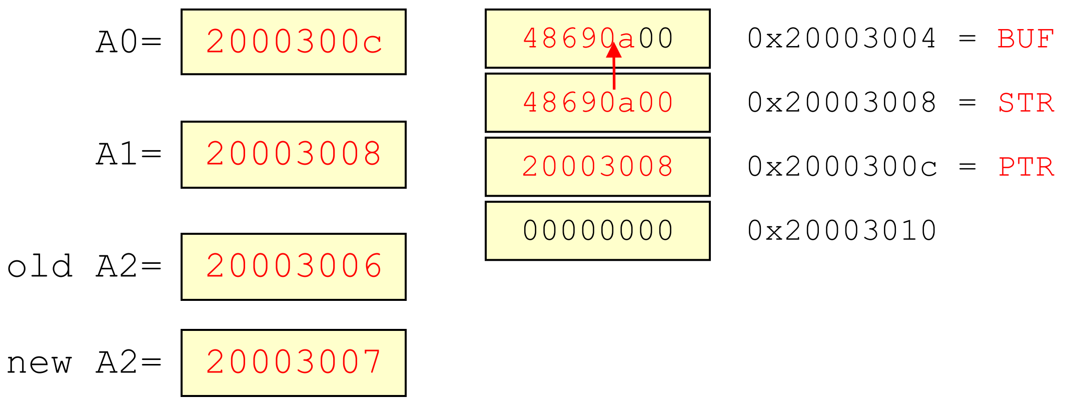
A0=	2000300c
A1=	20003008
A2=	20003006

48690000	0x20003004	= BUF
48690a00	0x20003008	= STR
20003008	0x2000300c	= PTR
00000000	0x20003010	

- BUF+2 evaluates to 0x20003004 + 2 = 0x20003006
- Since this IS an immediate (denoted by the '#' sign), we place that value, 0x20003006, in A2

Code Example 2

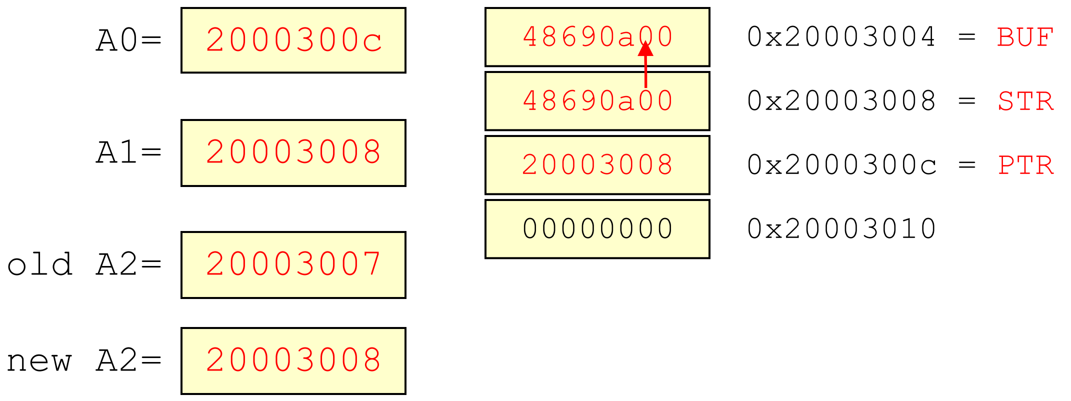
```
        .data       0x20003004
BUF:    .space      4
CNST    .equ        3
STR:    .asciz      'Hi\n'
PTR:    .long       BUF+4

        .text
MAIN:   MOVEA.L     #PTR,A0
        MOVE.W      STR,-8(A0)
        MOVEA.L     (A0),A1
        MOVEA.L     #BUF+2,A2
        MOVE.B      2(A1),(A2)+
        MOVE.B      CNST(A1),(A2)+
        STOP        #0x2700
```



- Take the address in A1 (which is 0x20003008) and add 2 to get the source data address = 0x2000300a
- Get the byte at 0x2000300a and place it at the address specified by A2 (i.e. 0x20003006)
- Then increment A2 by only 1 (since it is a byte operation)

Code Example 2

```
        .data     0x20003004
BUF:    .space    4
CNST    .equ      3
STR:    .asciz    'Hi\n'
PTR:    .long     BUF+4

        .text
MAIN:   MOVEA.L   #PTR,A0
        MOVE.W    STR,-8(A0)
        MOVEA.L   (A0),A1
        MOVEA.L   #BUF+2,A2
        MOVE.B    2(A1),(A2)+
        MOVE.B    CNST(A1),(A2)+
        STOP      #0x2700
```



- CNST is replaced by 3 earlier by the assembler…
- Take the address in A1 (which is 0x20003008) and add 3 to get the source data address = 0x2000300b
- Get the byte at 0x2000300b and place it at the address pointed to by A2
- Increment A2 by 1 (since .B)

Code Example 2

```
            .data      0x20003004
BUF:        .space     4
CNST        .equ       3
STR:        .asciz     'Hi\n'
PTR:        .long      BUF+4

            .text
MAIN:       MOVEA.L    #PTR,A0
            MOVE.W     STR,-8(A0)
            MOVEA.L    (A0),A1
            MOVEA.L    #BUF+2,A2
            MOVE.B     2(A1),(A2)+
            MOVE.B     CNST(A1),(A2)+
            STOP       #0x2700
```

A0=	2000300c
A1=	20003008
A2=	20003008

48690a00	0x20003004	= BUF
48690a00	0x20003008	= STR
20003008	0x2000300c	= PTR
00000000	0x20003010	

- After program execution the memory and register have the above contents