Example 1 – Problem Description

- Write a program to sum an array of numbers
  - Assumptions
    - Array is stored at location LIST
    - Numbers are bytes
    - Label SIZE indicates the length of the array
    - Sum is to be stored at the end of the array

Example 1 – Pseudo Code

- Partial C code

```
ptr = &LIST;
sum = 0;
counter = 0;

while(counter < SIZE){
    sum += *ptr++;
    counter++;
}
*ptr = sum;
```

Example 1 – Assembler Code

```
* A0(pointer to LIST), D1 (loop counter) D2(sum)

        movea.l     #list,a0        ;initialize ptr
        clr.l       d2              ;sum=0
        clr.l       d1              ;counter=0
loop    cmpi.b      #size,d1        ;last byte?
        bge         exit            ;yes, goto exit
        add.b       (a0)+,d2        ;add to sum
        addq        #1,d1           ;counter++
        bra         loop            ;do it again
exit    move.b      d2,(a0)         ;store sum

        org         $9000
list    dc.b        1,2,3           ;array of bytes
sum     ds.b        1               ;location of sum
size    equ         sum - list      ;array size
```

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```

MEMORY

Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 XXXXXXXX d1 XXXXXXXX d2 XXXXXXXX

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```

MEMORY

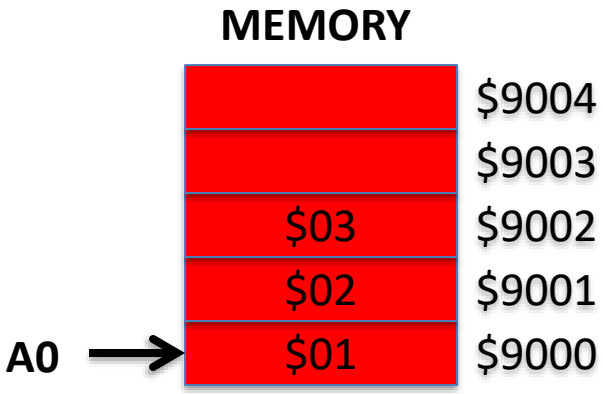
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 XXXXXXXX d1 XXXXXXXX d2 XXXXXXXX

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.l      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



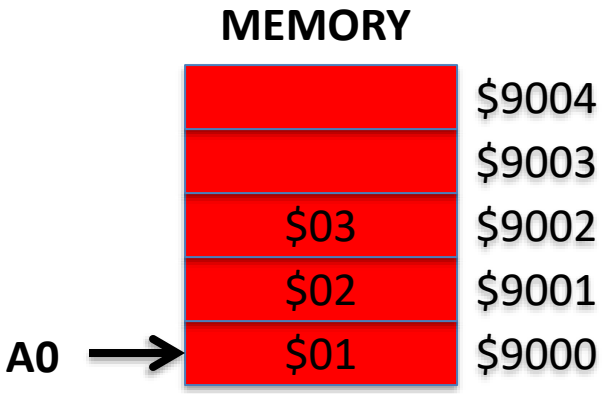
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009000 d1 XXXXXXXX d2 XXXXXXXX

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



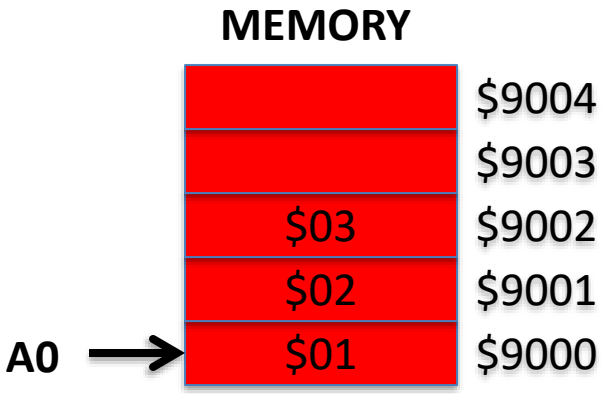
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009000 d1 XXXXXXXX d2 00000000

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



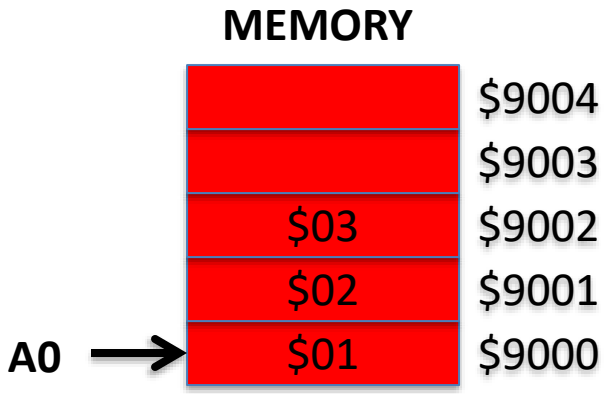
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009000 d1 00000000 d2 00000000

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



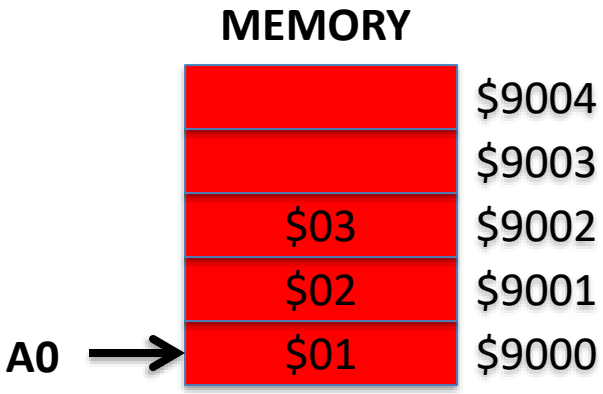
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009000 d1 00000000 d2 00000000

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



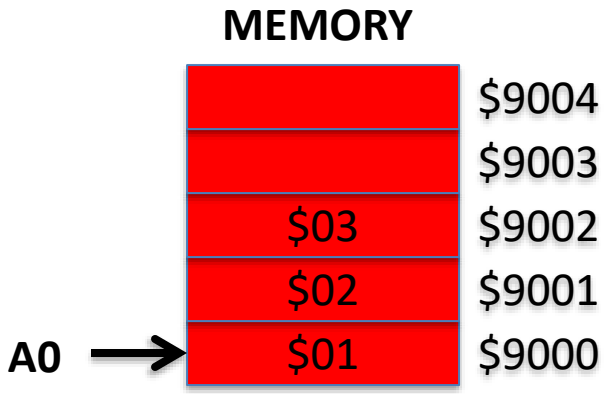
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009000 d1 00000000 d2 00000000

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



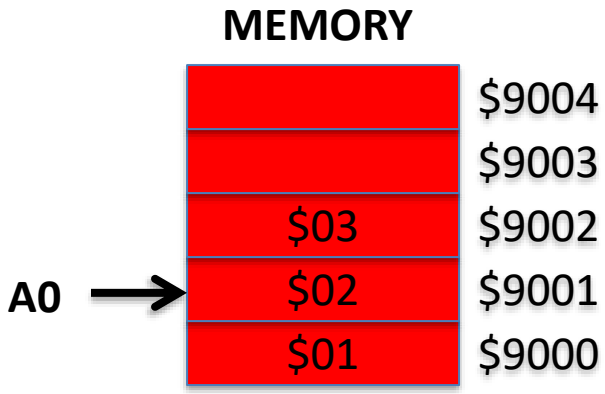
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009000 d1 00000000 d2 00000001

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



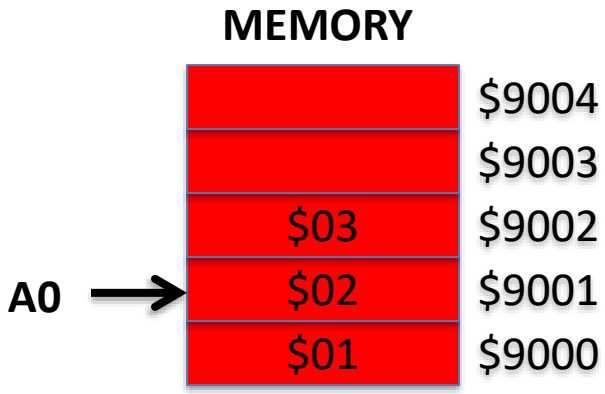
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009001 d1 00000000 d2 00000001

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



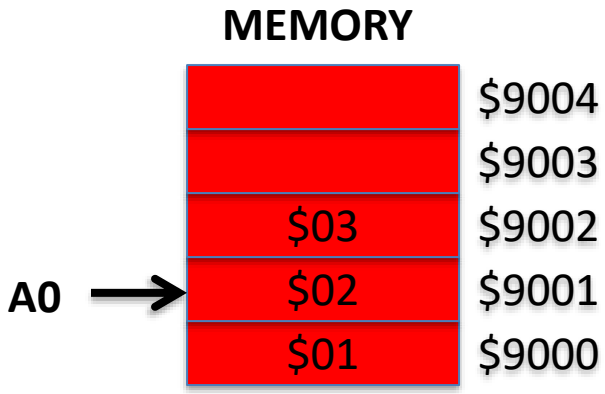
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009001 d1 00000001 d2 00000001

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



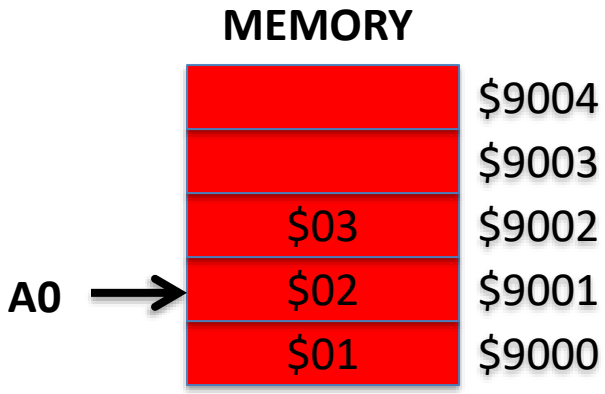
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009001 d1 00000001 d2 00000001

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



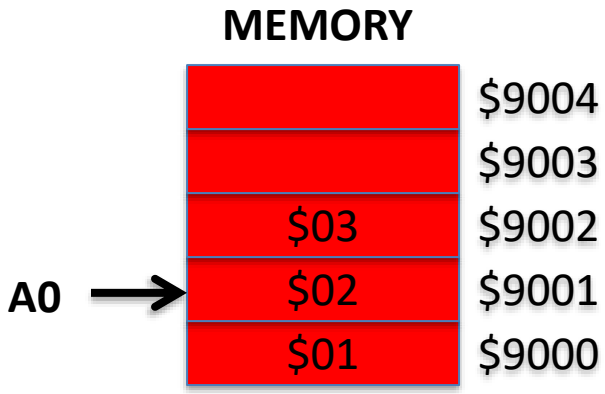
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009001 d1 00000001 d2 00000001

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



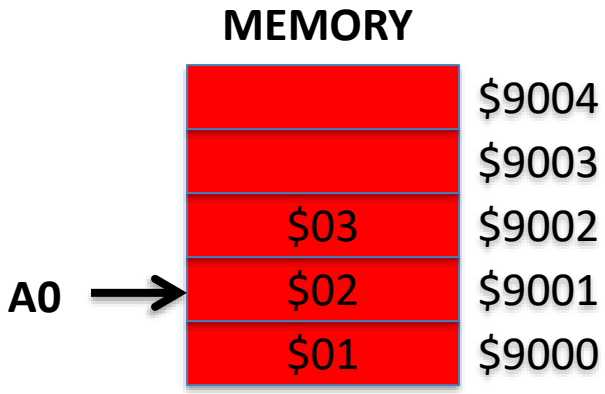
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009001 d1 00000001 d2 00000001

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



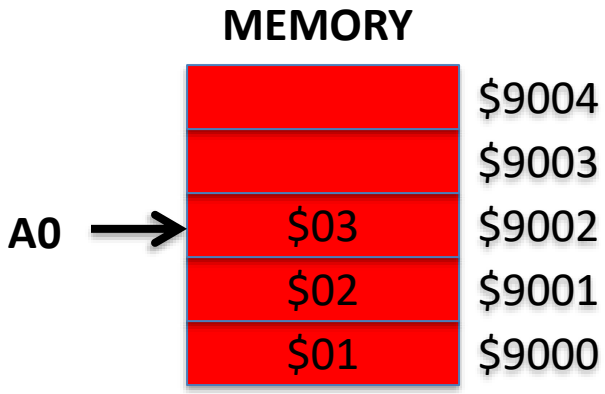
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009001 d1 00000001 d2 00000003

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



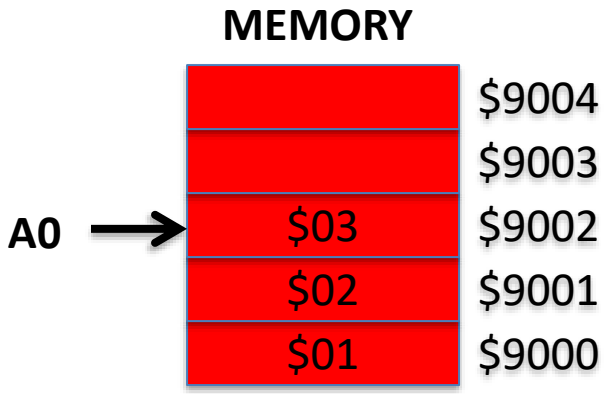
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009002 d1 00000001 d2 00000003

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



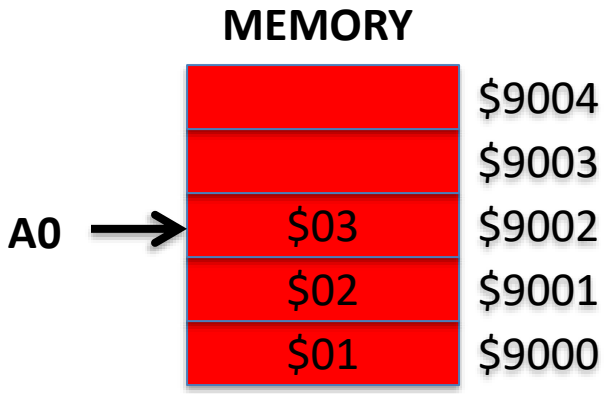
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009002 d1 00000001 d2 00000003

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



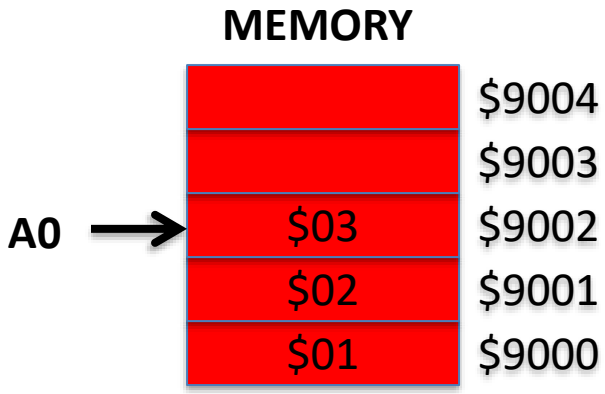
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009002 d1 00000002 d2 00000003

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



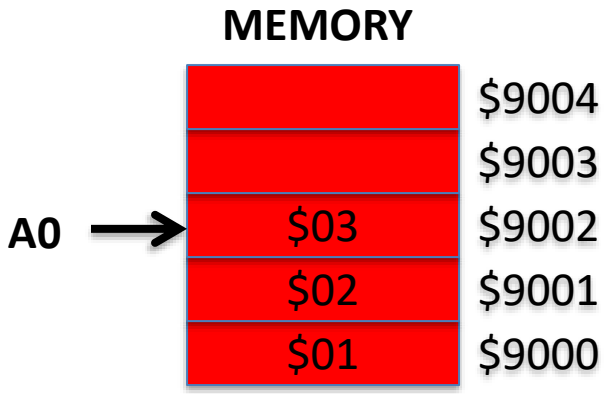
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009002 d1 00000002 d2 00000003

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



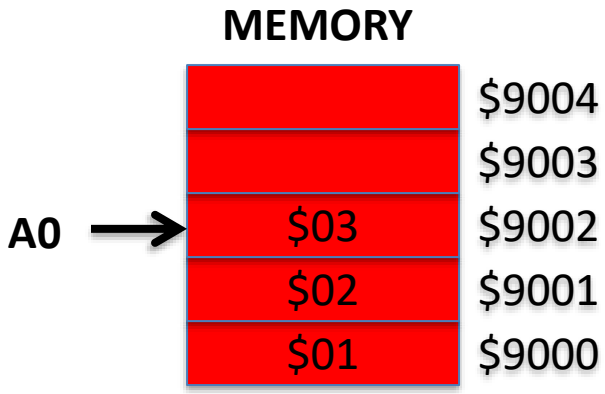
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009002 d1 00000002 d2 00000003

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



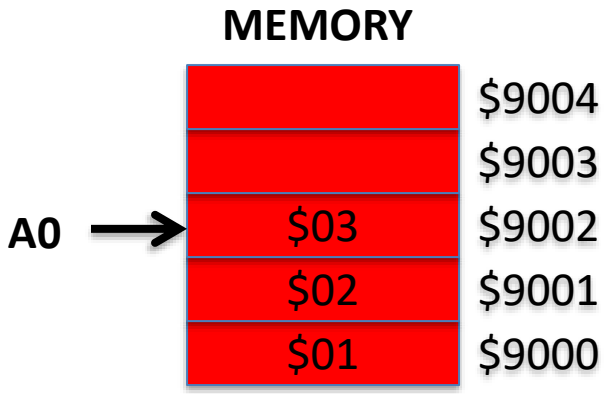
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009002 d1 00000002 d2 00000003

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)


        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



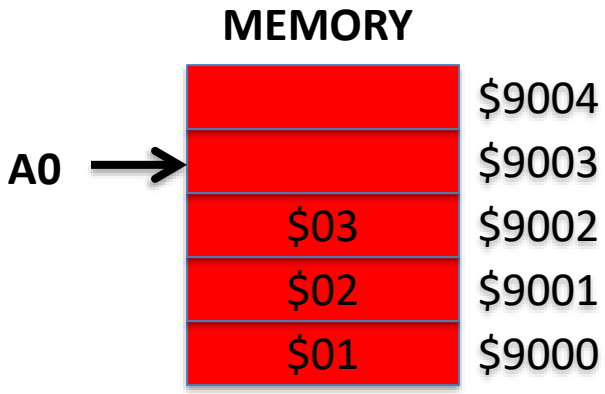
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009002 d1 00000002 d2 00000006

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



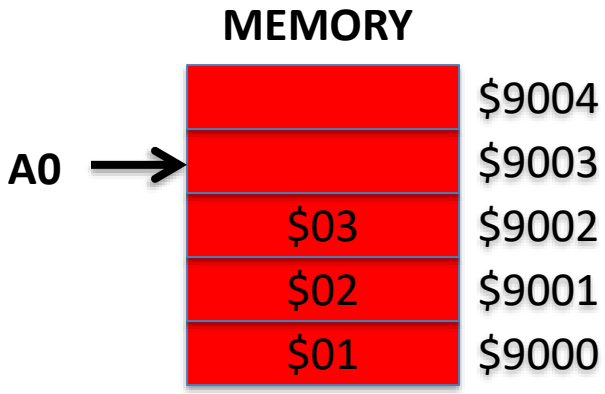
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009003 d1 00000002 d2 00000006

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



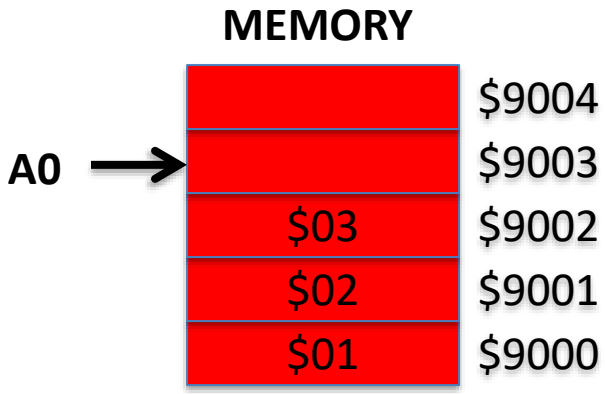
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009003 d1 00000002 d2 00000006

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



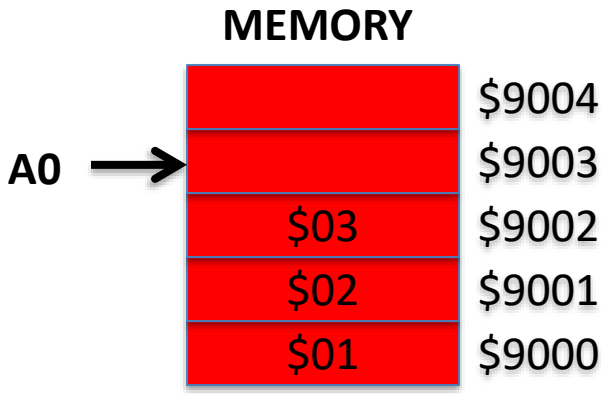
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009003 d1 00000003 d2 00000006

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



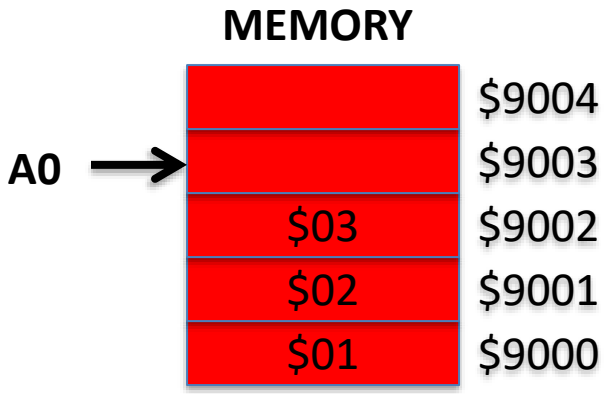
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009003 d1 00000003 d2 00000006

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



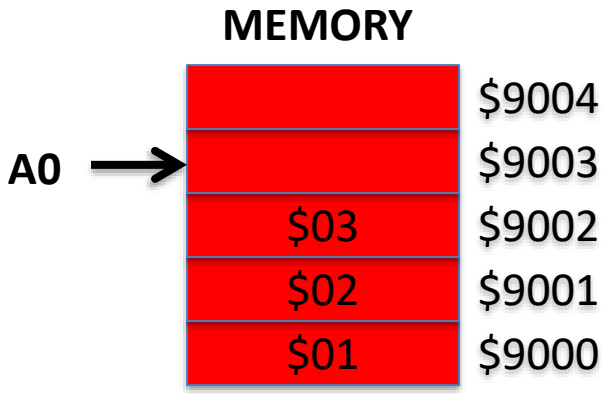
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009003 d1 00000003 d2 00000006

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



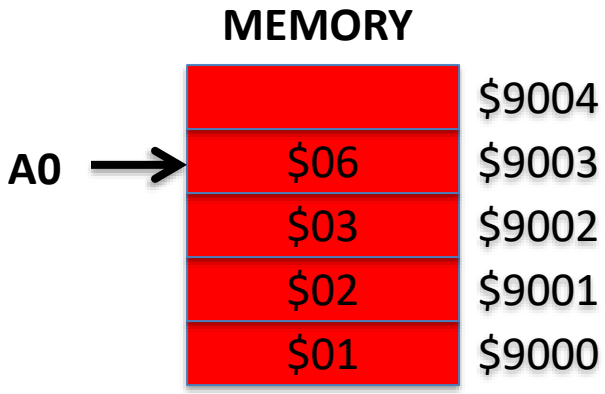
Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009003 d1 00000003 d2 00000006

Example 1 – Trace

```
        movea.l     #list,a0
        clr.l       d2
        clr.l       d1
loop    cmpi.b      #size,d1
        bge         exit
        add.b       (a0)+,d2
        addq        #1,d1
        bra         loop
exit    move.b      d2,(a0)

        org         $9000
list    dc.b        1,2,3
sum     ds.b        1
size    equ         sum - list
```



Symbol Table

Symbol	Value
list	00009000
sum	00009003
size	00000003

a0 00009003 d1 00000003 d2 00000006

Example 2 – Problem Description

- Write a program to count the number of (strictly) positive numbers in an array of numbers
  - Assumptions
    - Array is stored at location LIST
    - Numbers are bytes
    - Label SIZE indicates the length of the array
    - The count is to be stored at the end of the array

Example 2 – Pseudo Code

- Partial C code

```
ptr = &LIST;
positive = 0;
counter = 0;

while(counter < SIZE){
   if(*ptr++ > 0)
      positive++;
   counter++;
}
*ptr = positive;
```

Example 2 – Assembler Code

```
* A0(pointer to LIST), D1 (loop counter), D2(positive)
        movea.l     #list,a0      ;initialize ptr
        clr.b       d2            ;positive=0
        clr.b       d1            ;counter=0
loop    cmpi.b      #size,d1      ;last byte?
        bge         exit          ;yes, goto exit
        cmpi.b      #0,(a0)+      ;if byte not positive
        ble         cont          ;then don't count
        addq        #1,d2         ;add to positive
cont    addq        #1,d1         ;counter++
        bra         loop          ;do it again
exit    move.b      d2,(a0)       ;store positive
                                  ;return to 68KMB
        org         $9000
list    dc.b        1,-2,-3       ;array of bytes
pos     ds.b        1             ;location of positive
size    equ         pos - list    ;array size
```

Testing a Single Operand

TST Test an Operand

Syntax: `TST <ea>`

Operation: CCR ← operand - zero

Dn	An	(An)	(An)+	-(An)	d(An)	d(An,Xn)	ABS.W	ABS.L	Imm
✓		✓	✓	✓	✓	✓	✓	✓	

Example 2 – Assembler Code Revisited

```
* A0(pointer to LIST), D1 (loop counter), D2(positive)
        movea.l     #list,a0      ;initialize ptr
        move.b      d2            ;positive=0
        move.b      d1            ;counter=0
loop    cmpi.b      #size,d1      ;last byte?
        bge         exit          ;yes, goto exit
        tst.b       (a0)+         ;byte negative or zero?
        ble         cont          ;then don't count
        addq        #1,d2         ;add to positive
cont    addq        #1,d1         ;counter++
        bra         loop          ;do it again
exit    move.b      d2,(a0)       ;store sum
                                  ;return to 68KMB
        org         $9000
list    dc.b        1,-2,-3       ;array of bytes
pos     ds.b        1             ;location of positive
size    equ         pos - list    ;array size
```

Example 3

- How would you modify the previous program if the numbers in the array were unsigned and you wanted to find those with values greater than 127?

Example 3

- How would you modify the previous program if the numbers in the array were unsigned and you wanted to find those with values greater than 127?

```
ptr = &LIST;
counter = 0;
greater = 0;

while(counter < SIZE){
   if(*ptr++ > 127)
      greater++;
   counter++;
}
*ptr = greater;
```

Example 3 – Assembler Code

```
* A0(pointer to LIST), D1 (loop counter) D2(greater)
        movea.l     #list,a0      ;initialize ptr
        move.b      d2            ;positive=0
        move.b      d1            ;counter=0
loop    cmpi.b      #size,d1      ;last byte?
        bge         exit          ;yes, goto exit
        cmpi.b      #127,(A0)+    ;byte <= 127?
        bls         cont          ;then don't count
        addq        #1,d2         ;add to greater
cont    addq        #1,d1         ;counter++
        bra         loop          ;do it again
exit    move.b      d2,(a0)       ;store sum
                                  ;return to 68KMB
        org         $9000
list    dc.b        1,-2,-3       ;array of bytes
grt     ds.b        1             ;location of greater
size    equ         grt - list    ;array size
```

Example 4

- How would you modify the previous program if the array contained characters and you wanted to count the number of upper-case alphabetical characters?

Example 4

- How would you modify the previous program if the array contained characters and you wanted to count the number of upper-case alphabetical characters?

```
ptr = LIST;
counter = 0;
upper = 0;

while(counter < SIZE){
   if(*ptr >= 'A' && *ptr <= 'Z')
      upper++;
   ptr++
   counter++;
}
*ptr = upper;
```

Example 4 – Assembler Code

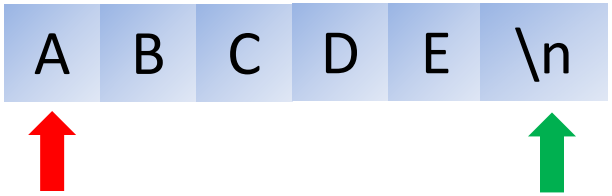
```
* A0(pointer to LIST), D1 (loop counter) D2(upper)
        movea.l     #list,a0      ;initialize ptr
        move.b      d2            ;positive=0
        move.b      d1            ;counter=0
loop    cmpi.b      #size,d1      ;last byte?
        bge         exit          ;yes, goto exit
        cmpi.b      #'A',(A0)     ;byte before 'A'?
        blo         cont          ;then don't count
        cmpi.b      #'Z',(A0)     ;byte after 'Z'?
        bhi         cont          ;then don't count
        addq        #1,d2         ;add to upper
cont    addq        #1,d1         ;counter++
        adda.l      #1,a0         ;point to next char
        bra         loop          ;do it again
exit    move.b      d2,(a0)       ;store sum
                                  ;return to 68KMB
* data does not change
```

Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available

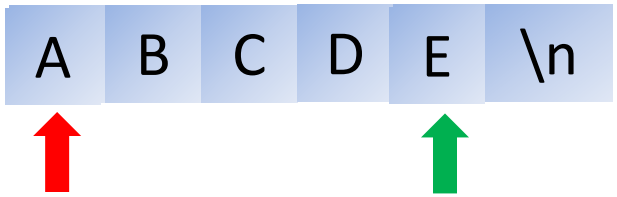
Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available



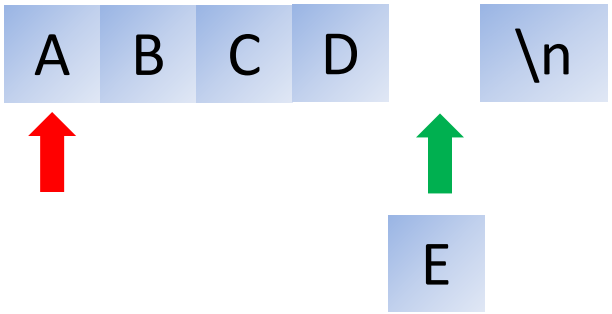
Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available



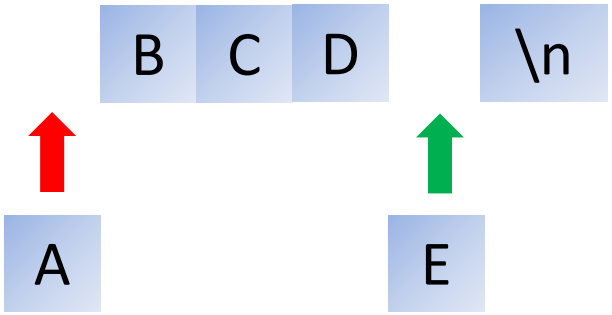
Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available



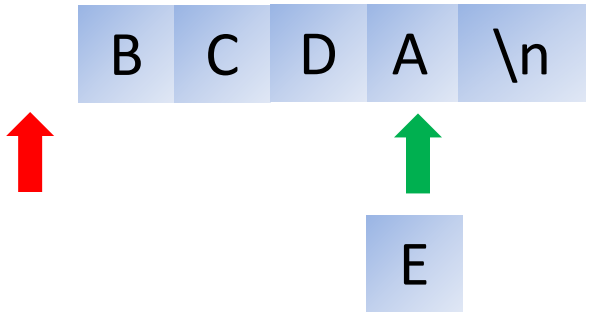
Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available



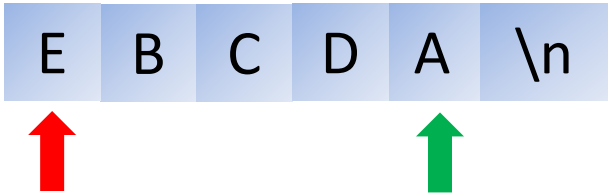
Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available



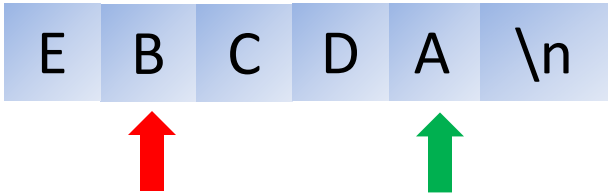
Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available



Example 5 – Problem Description

- Write a program to reverse the characters in a character array that is terminated with a carriage return
  - Assumption
    - No additional memory is available



Example 5 – Assembler Code

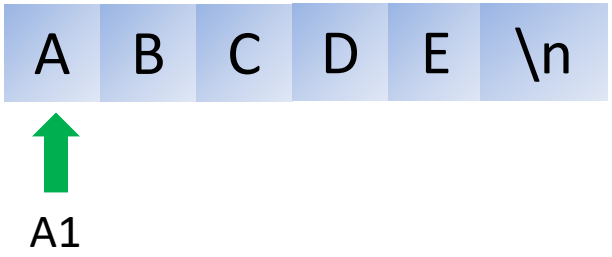
```
* A0/A1(ptrs to start/end of string), D0 (temporary)

* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no: try again
        suba.l      #1,a1         ;backup one byte
* Reverse characters
        movea.l     #string,a0    ;point to start of string
loop2   cmpa.l      a0,a1         ;if A1 <= A0
        bls         exit          ;then string is reversed
        move.b      (a0),d0       ;otherwise
        move.b      -(a1),(a0)+   ;swap
        move.b      d0,(a1)       ;bytes
        bra         loop2         ;do it again!
exit                              ;return to 68KMB
string  dc.b        'ABCDE',$D    ;string to reverse
```

Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```

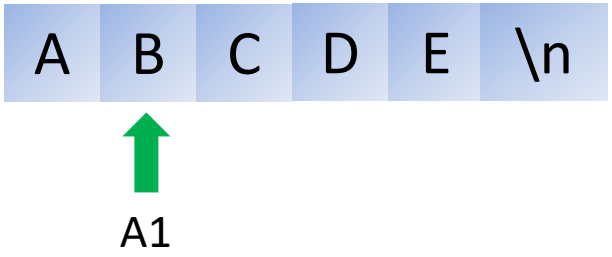
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



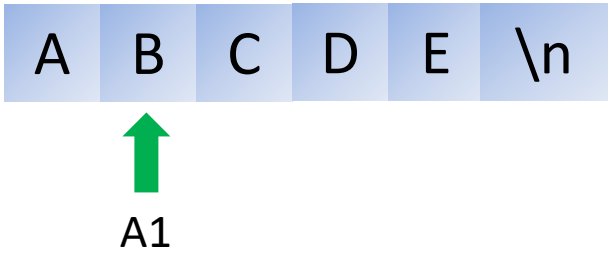
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



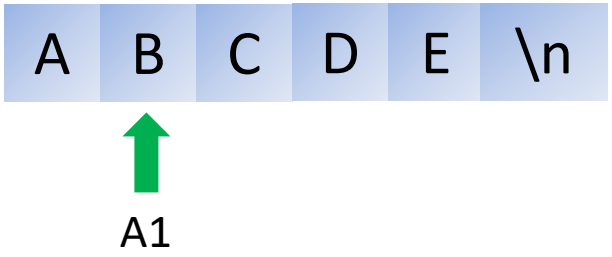
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



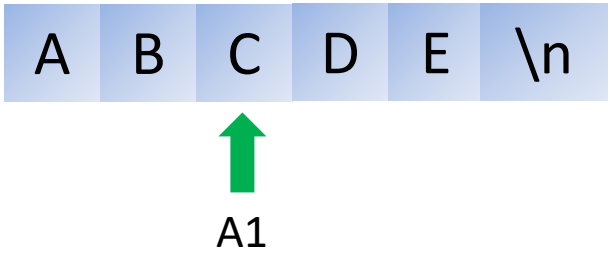
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



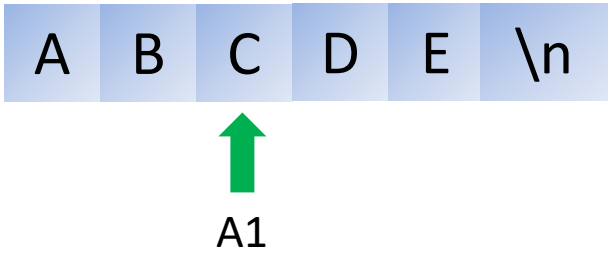
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



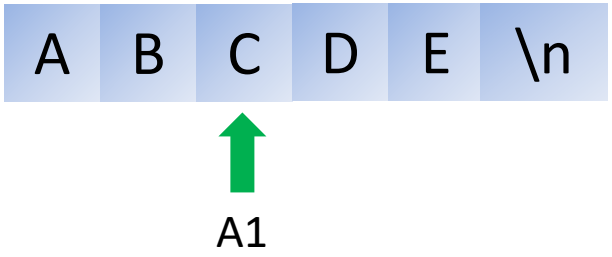
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



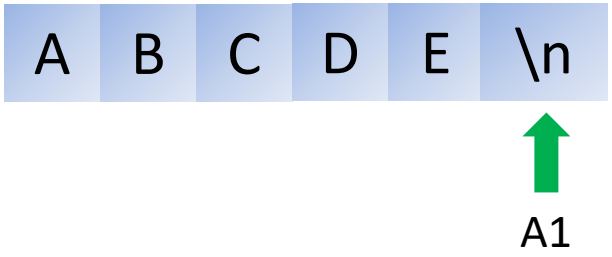
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



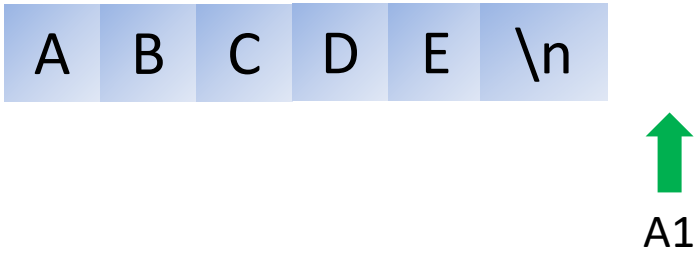
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



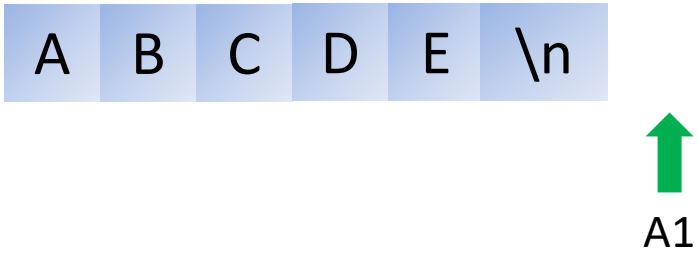
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



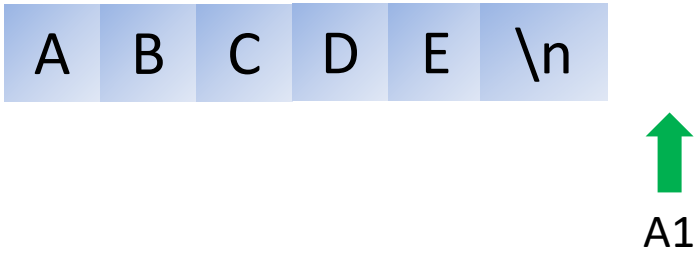
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



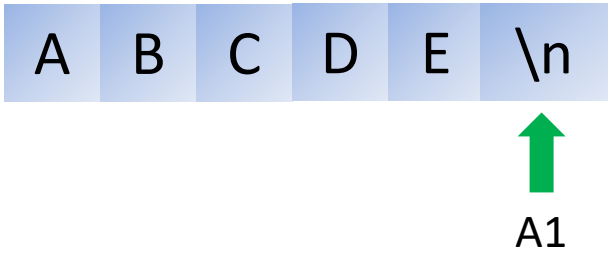
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



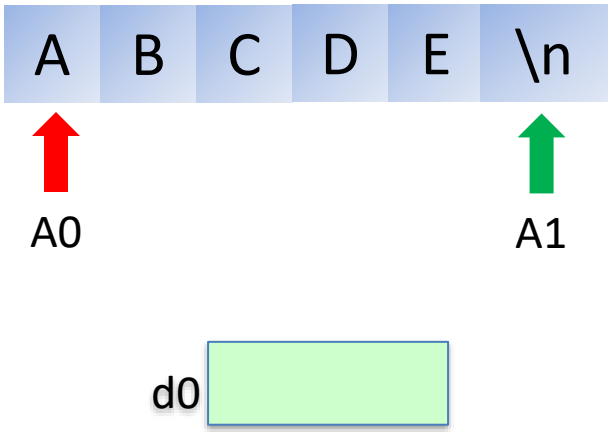
Example 5 – Determining End of String

```
* Position ptr (a1) at end of string
        movea.l     #string,a1    ;point to start of string
loop1   cmpi.b      #$D,(a1)+     ;carriage return?
        bne         loop1         ;no(Z=0): try again
        suba.l      #1,a1         ;backup one byte
```



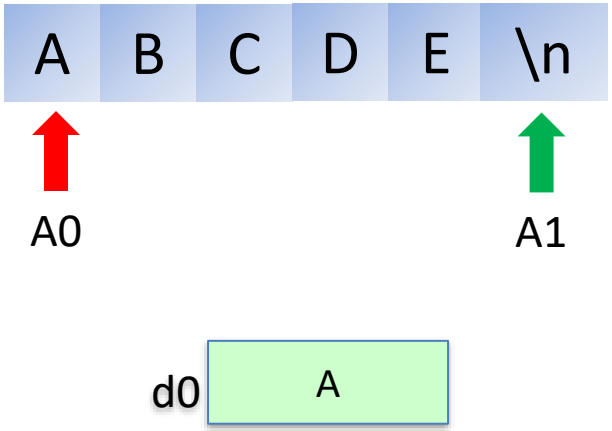
Example 5 – Swapping Characters

```
* swap pair of chracters
        move.b      (a0),d0         ;save bottom char
        move.b      -(a1),(a0)+     ;move top char to bottom
        move.b      d0,(a1)         ;move bottom char to top
```



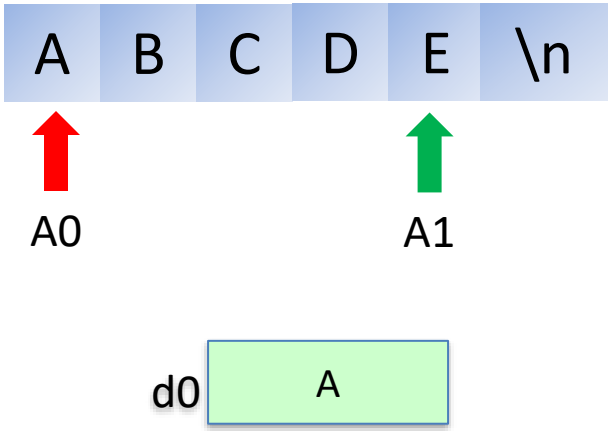
Example 5 – Swapping Characters

```
* swap pair of chracters
        move.b      (a0),d0         ;save bottom char
        move.b      -(a1),(a0)+     ;move top char to bottom
        move.b      d0,(a1)         ;move bottom char to top
```



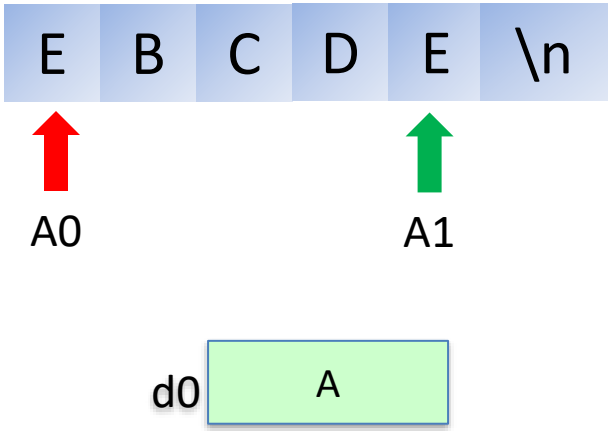
Example 5 – Swapping Characters

```
* swap pair of chracters
        move.b      (a0),d0         ;save bottom char
        move.b      -(a1),(a0)+     ;move top char to bottom
        move.b      d0,(a1)         ;move bottom char to top
```



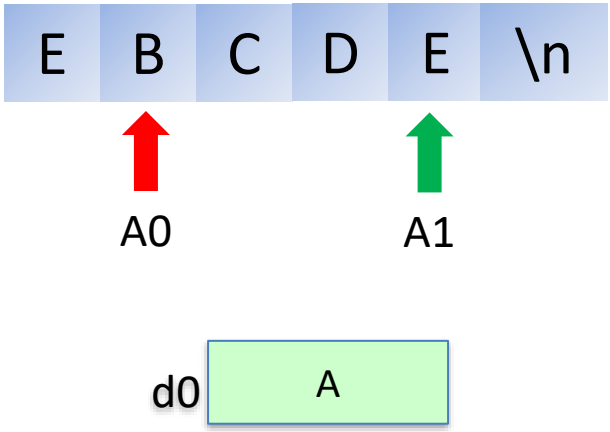
Example 5 – Swapping Characters

```
* swap pair of chracters
        move.b      (a0),d0         ;save bottom char
        move.b      -(a1),(a0)+     ;move top char to bottom
        move.b      d0,(a1)         ;move bottom char to top
```



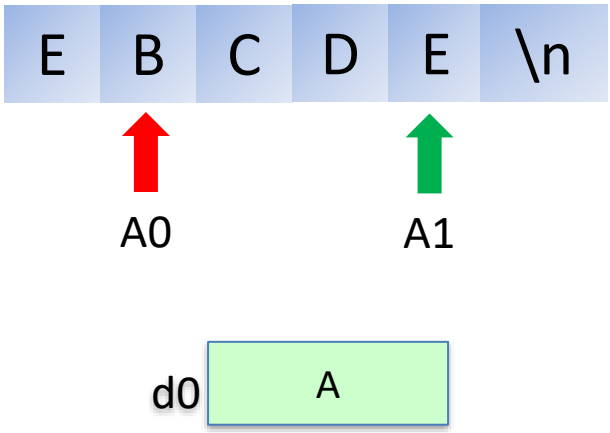
Example 5 – Swapping Characters

```
* swap pair of chracters
        move.b      (a0),d0         ;save bottom char
        move.b      -(a1),(a0)+     ;move top char to bottom
        move.b      d0,(a1)         ;move bottom char to top
```



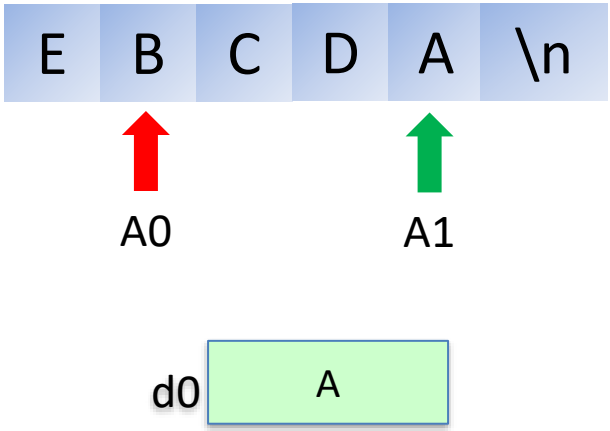
Example 5 – Swapping Characters

```
* swap pair of chracters
        move.b      (a0),d0         ;save bottom char
        move.b      -(a1),(a0)+     ;move top char to bottom
        move.b      d0,(a1)         ;move bottom char to top
```



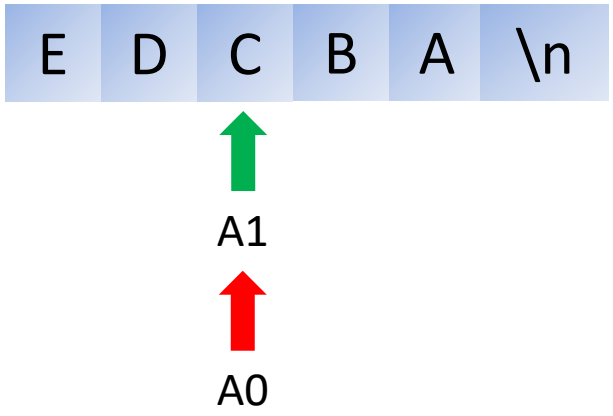
Example 5 – Swapping Characters

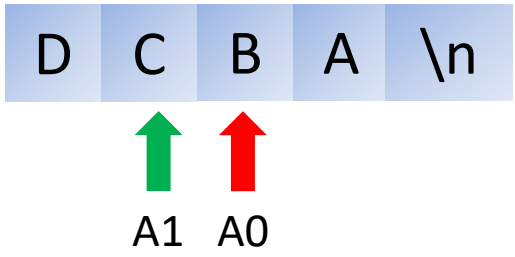
```
* swap pair of chracters
        move.b      (a0),d0         ;save bottom char
        move.b      -(a1),(a0)+     ;move top char to bottom
        move.b      d0,(a1)         ;move bottom char to top
```



Example 5 – When is the String Reversed?

```
* Loop exit condition
  loop2         cmpa.l   a0,a1        ;if A1 <= A0
                bls      exit         ;then string is reversed
```





Summary

- Use unsigned branches when comparing
  - Unsigned integers
  - Characters
  - Addresses
- Only used signed branches when comparing 2's complement integers
- Use CMPA when comparing pointers
- Use CMPI when comparing a constant to a value
- Be careful when combining pointer arithmetic with short-circuit evaluation