

Array Example: Memory Layout

- the memory layout of the previous (and next) example is

a:	'a'	a[0]
	'b'	a[1]
	'c'	a[2]
	'd'	a[3]
	'\0'	a[4]
b:		b[0]
		b[1]
	5	b[2]
		b[3]
i:	2	

- general method: put the value of the index in XR (via AC), then simply use indexed addressing mode
- strings: position of the first NULL ('`\0`') character defines the length
 - can be more conveniently initialized using the `data` directive
 - e.g. `a: data "abcd\0"`

Arrays: Further Translation Patterns

- array access inside conditions: just set up XR before evaluating

```

load    i          ;   if (b[i] > 0) {
storexr ;
load    *b         ;                               /*AC = mem[b+XR]*/
cmp     #0         ;
ble     endif1    ;
loadxr x          ;   x = x + b[i]; /*AC = mem[x]*/
add     *b         ;                               /*AC += mem[b+XR]*/
store  x          ;   }
endif1: ;

```

- one can use XR as an index variable: (c.f. array-example.ass)

```

load    #0         ;   i = 0;           /*AC = 0*/
storexr ;                               /*XR = AC*/
repeat1: ;   do {
load    *a+1      ;   b[i] = a[i+1];
store  *b         ;
incxr  #1         ;   i = i+1;
loadxr ;                               /*AC = XR*/
cmp    #N         ;   } while (i != N);
bne   repeat1    ;

```

Further Array Example: array-example.ass

```

N      = 4        ;   #define N 4;
a:     block     N      ;   char a[N+1];
       block     1      ;
b:     block     N      ;   int b[N];
i:     block     1      ;   int i;
       ;

...    ; // (code to initialize a[] omitted)
load   #0        ;   i = 0;
store  i         ;
repeat1: ;   do {
load   i         ;   b[i] = a[i+1]; /* AC = mem[i] */
storexr ;                               /* XR = AC */
load   *a+1     ;   /* AC = mem[a+1+XR] */
store  *b       ;   /* mem[b+XR] = AC */
load   i         ;   i = i+1;
add    #1       ;
store  i         ;
load   #N       ;   } while (i < N);
cmp    i        ;
bgt   repeat1   ;

```

Arrays in Assembly Language - Review

- more difficult if two different indices are used:
 - e.g. `b[j] = a[i] + 42;`
 - solution?
- define an array using `block` `array_length`
- set XR to value of the index (usually start at 0)
- reduce access of multi-dimensional array elements to an effective 1-dimensional access (e.g. through row-major ordering) (*...later*)
- for arrays, the machine needs an indexed addressing mode to efficiently access individual elements

