Pointers and Arrays in C and Assembly Language
Arrays

• Provide useful abstraction
• Match up to machine memory organization
• Array index computation
  – time consuming
  – is there a faster way
Arrays and Pointers in C

- Arrays can also be accessed with pointers in C
- Uses low-level model of memory at the high-level language level
- Can result in faster code
  - but modern compilers can often do this work, so we are better off, in general, using the high-level abstractions
C code examples - Set array values to 0

// array version
clear1(int array[], int size)
{ int i;
    for(i = 0; i < size; i = i+1)
        array[i] = 0;
}

// pointer version
clear2(int *array, int size)
{ int *p;
    for(p = &array[0]; p < &array[size]; p = p+1)
        *p = 0;
}
Array version assembly code

clear1:
   add  $t0, $zero, $zero  # i = 0
fori:   sll  $t1, $t0, 2        # $t1 = 4*i
   add  $t1, $a0, $t1      # $t1 = addr of array[i]
   sw   $zero, 0($t2)      # array[i] = 0
   addi $t0, $t0, 1        # i = i + 1
   slt  $t2, $t0, $a1      # i < size ?
   bne  $t2, $zero, fori   # if so, continue
endfor:

Six instructions in loop.
clear2:
    add $t0, $a0, $zero    # p = addr of array[0]
forp:   sw   $zero, 0($t0)      # memory[p] = 0
    addi $t0, $t0, 4        # p = p + 4
    sll  $t1, $a1, 2        # $t1 = 4 * size
    add  $t1, $t1, $a0      # $t1 = addr array[size]
    slt  $t2, $t0, $t1      # p < &array[size] ?
    bne  $t2, $zero, forp   # if so, continue
endfor:
Pointer version assembly code

clear2:
    add $t0, $a0, $zero       # p = addr of array[0]
    sll  $t1, $a1, 2         # $t1 = 4 * size
    add  $t1, $t1, $a0       # $t1 = addr array[size]
forp:   sw   $zero, 0($t0)      # memory[p] = 0
    addi $t0, $t0, 4        # p = p + 4
    slt  $t2, $t0, $t1      # p < &array[size] ?
    bne  $t2, $zero, forp   # if so, continue
endfor:

Four instructions in loop
What to use in code

• 1970's
  – compilers simpler
  – optimize in C code

• 1990’s and beyond
  – optimizing compilers can convert code for arrays to be similar to pointer code
  – Programming in high level language should use the appropriate abstraction
    • let the compiler do the work
    • fewer errors in program using appropriate abstractions