Pointers

CSE 361S

Pointers in C

• A pointer is a variable that contains the address of a variable

• Major uses of pointers:
  – Direct access to specific memory location (address)
  – Manipulation of array elements, especially multi-dimensional arrays
  – Program management of dynamically allocated memory
  – Passing arguments by reference

• Indirect addressing modes in assembly language represent exactly the same concept.

Pointer size is machine dependent

• 32 bits on x86 ISA we are using in class

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0007f80</td>
<td>0x0007f80</td>
<td>0x0007f80</td>
<td>0x0007f80</td>
</tr>
<tr>
<td>0x00083a0</td>
<td>0x00083a0</td>
<td>0x00083a0</td>
<td>0x00083a0</td>
</tr>
<tr>
<td>char * p &amp;c</td>
<td>Data</td>
<td>Data</td>
<td>Data</td>
</tr>
<tr>
<td>&amp;c gives address of variable c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;p gives address of variable p</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pointer Declaration

[type] * [name] {= [init val];}

• e.g.,
  char c = 'A';
  char * p = &c;
  & is operator that returns address of a variable

• e.g.,
  – &c gives address of variable c
  – &p gives address of variable p

Pointer Use

* is the “indirection” or “dereference” operator, when applied to a pointer it accesses the memory at the address stored in the pointer variable.

* operator can exist on either side of assignment:
  char c = 'A';
  char * p = &c;
  char fred;
  fred = * p;
  fred = 'A';
  * p = 'B';
  c = 'A';

Array Names and Pointers

• Note: name of array is equivalent to pointer to first element

• e.g.,
  int z[10];
  z is the same thing as &z[0]
  char z;
  char &z[0];
  char z[0];
Example Use

```c
int x = 1;
int y = 2;
int z[10];
int *ip;
int *iq;
```

Physical Address Example

```c
unsigned int * status;

status = (unsigned int *)0x8000040c;
for (i=0; i<24; i++) {
    mask = 1 << i;
    status_bit[i] = *status & mask;
    status_bit[i] >>= i;
}
```

Pointers and Function Arguments

```c
void swap (int x, int y) {
    int temp;
    temp = x;
    x = y;
    y = temp;
}

swap (a, b);
```

• Note: C passes parameters a and b by “value”, so above call to swap does not change a and b, only copies of a and b.
Working Pass by Reference

```c
void swap (int * px, int * py) {
    int temp;
    temp = *px;
    *px = *py;
    *py = temp;
}
swap (&a, &b);
```

- WWU