Pointers

A pointer is a variable that contains the address of some data object.

Major uses:
- direct access to specific mem location
- manipulation of array elements
- prog. management of dynamically alloc. mem.
- passing arguments by reference

In assembly lang., indirect addressing mode represents exactly same concept

Size of pointer is machine dependent
(32 bits on x86 ISA used in class)

```c
short int var1 0x7fa8
char var1 0x7faa
char var3 0x7fab
...
char * ptr2 0xaaa 0x7f to var2
```

Declaring pointer
```
[ type ] * [ name ] { = [init val] };
```

E.g., char c = 'A';
char & p = & c;

& is operator that returns address of variable.
& c address of c
& p address of p
X is the "indirection" or "dereference" operator. When applied to a pointer it accesses the "pointed-to" memory.

X can go on either side of assignment.

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

int x[10]; z = &x[0]

int x = 1;
int y = 2;
int z = 10;
int *ip;
int *ip;

ip = &x;  // ip points to x. */
y = *ip;   // *ip accesses contents of x. */
*ip = 0;   // *ip sets x to 0. */
ip = &x[0]; // ip now points to first element.
*ip = 13;  // x[0] = 13. */
*ip = 13;  // &x[0] = &x[2]. */
*ip = *ip + 10; // adds 10 to &x[0]. */
*ip = 1;   // equiv. to (*ip)--
*ip = 1;   // not (*ip)-- */
q = *ip;   //
\[(x + y)^{1/4}\] \(x = z + 20 \text{ now } 2z + x\]

\[i = i + 1; \quad \text{if no points to } z + 1\]

\[x, y = 0; \quad \text{if } z + 1 \leq 0 \times 1\]

\[x = y + 1; \quad \text{if } x \text{ new points to } z + 1\]

\[i = i + 1; \quad \text{if } i \text{ new points to } z + 1\]

\[x, y = x, y + 1; \quad \text{if } x \text{ or } y \text{ new points to } z + 2\]

\[i = i + 1; \quad \text{if } i \text{ new points to } z + 2\]

\[x, y = \text{anything } \times \text{ BAD, BAD, BAD!} \times\]

---

**Physical code example**

```c
unsigned int & stat;

status = (unsigned int) 0x800040c;

for (i = 0; i < 24; i++) {
    mask = 1 << i;
    stat_bit(i) = (stat) & mask;
}
```

---

**Function arguments**

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

C passes parameters `a`, `b` by value.
void swap (int *px, int *py);
int temp;
    temp = *px;
    *px = *py;
    *py = temp;

swap (&a, &b);

The call is by reference

command line arguments in C

int main (int argc, char * argv[ar])

argc is number of arguments (incl. program)
argv is array of strings
argv[0] : program name
argv[argc-1] : last arg.

int main (int argc, char * argv[ar]) {
    int a,
    printf ("%d arg: argv[i] =%s\n", argc, argv[i]);
    for (i=0; i < argc; i++) {
        printf (" %d: %s\n", i, argv[i]);
    }
    return 0;
}

e改变.
2. /echo args
   Hi there

3. args

0: /echo args
1: Hi
2: then

7 args c -o echo args