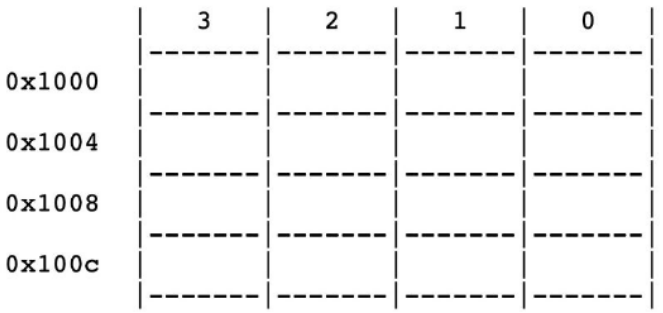
Your machine has four byte pointers and is little endian.

memcpy(dest, src, size) copies **size** bytes from **src** to **dest**.

The program to the right has ten lines of output.

- **Question 1**: What is line 1 of the output?
- **Question 2**: What is line 2 of the output?
- **Question 3**: What is line 3 of the output?
- **Question 4**: What is line 4 of the output?
- **Question 5**: What is line 5 of the output?
- **Question 6**: What is line 6 of the output?
- **Question 7**: What is line 7 of the output?
- **Question 8**: What is line 8 of the output?
- **Question 9**: What is line 9 of the output?
- **Question 10**: What is line 10 of the output?

Help: Suppose k = 0x1000. Then use this drawing:




```
int main()
{
  unsigned int k[4];
  unsigned char *cp;
  unsigned int *ip;
  int i;

  cp = (unsigned char *) k;
  ip = k+2;
  for (i = 0; i < 16; i++) {
    cp[i] = i*16 + 15-i;
  }

  printf("0x%x\n", cp[3]);    /* Output line 1. */
  printf("0x%x\n", cp[7]);    /* Output line 2. */
  printf("0x%x\n", k[0]);     /* Output line 3. */
  printf("0x%x\n", *ip);      /* Output line 4. */

  memcpy(cp+2, cp+10, 4);
  printf("0x%x\n", cp[3]);     /* Output line 5. */
  printf("0x%x\n", cp[4]);     /* Output line 6. */
  printf("0x%x\n", k[0]);      /* Output line 7. */
  printf("0x%x\n", k[1]);      /* Output line 8. */

  cp += 12;
  ip = (unsigned int *) cp;
  i = ip - k;
  printf("%d\n", i);           /* Output line 9. */

  i = cp - (unsigned char *) k;
  printf("%d\n", i);           /* Output line 10. */
  return 0;
}
```

Clicker Questions -- Answers

From the first few lines, there are only 16 bytes in question -- k[0] through k[3]. Let's label them before they are set:

```
              3      2      1      0
      k[0]                           cp[0] through cp[3]
      k[1]                           cp[4] through cp[7]
ip[0] k[2]                           cp[8] through cp[11]
ip[1] k[3]                           cp[12] through cp[15]
```

The **for** loop sets the bytes. Each **cp[i]** will be of the form `0xwy`, where *w* is equal to *i*, and *y* is equal to 15-*i*. Here is what the 16 bytes look like:

```
                3      2      1      0
               ----   ----   ----   ----
      k[0]     0x3c   0x2d   0x1e   0x0f     cp[0] through cp[3]
      k[1]     0x78   0x69   0x5a   0x4b     cp[4] through cp[7]
ip[0] k[2]     0xb4   0xa5   0x96   0x87     cp[8] through cp[11]
ip[1] k[3]     0xf0   0xe1   0xd2   0xc3     cp[12] through cp[15]
```

This allows us to answer the first four questions:

- **Question 1**: `0x3c`
- **Question 2**: `0x78`
- **Question 3**: `0x3c2d1e0f`
- **Question 4**: `0xb4a59687`

The `memcpy()` statement will move the four bytes starting at index cp+10 to the four bytes starting at index cp+2. When it's done, here are the 16 bytes of memory:

```
                3      2      1      0
               ----   ----   ----   ----
      k[0]     0xb4   0xa5   0x1e   0x0f     cp[0] through cp[3]
      k[1]     0x78   0x69   0xd2   0xc3     cp[4] through cp[7]
ip[0] k[2]     0xb4   0xa5   0x96   0x87     cp[8] through cp[11]
ip[1] k[3]     0xf0   0xe1   0xd2   0xc3     cp[12] through cp[15]
```

This allows us to answer the next four questions:

- **Question 5**: `0xb4`
- **Question 6**: `0xc3`
- **Question 7**: `0xb4a51e0f`
- **Question 8**: `0x7869d2c3`

`cp` and `ip` are now set to be `(k+3)`. So the answer to question 9 is three.

When we do the pointer arithmetic by bytes rather than ints, the difference between `cp` and `k` is 12, so the answer to question 10 is 12.