# Why?

You know how to represent various kinds of data as bits, but how do we refer to individual data items in the memory of the computer?

# Model 1: Bytes and addresses

Two views of the same memory. One organized by bytes and one by words.

|  |  |  |
| --- | --- | --- |
| address | memory contents(by byte) | memory contents(by word) |
| 0x00000000 | 0x00 | (will fill this box in #8) |
| 0x00000001 | 0x15 |
| 0x00000002 | 0x03 |
| 0x00000003 | 0x1A |
| 0x00000004 | 0x99 | 0xBF004199 |
| 0x00000005 | 0x41 |
| 0x00000006 | 0x00 |
| 0x00000007 | 0xBF |
| 0x00000008 | 0x00 | 0x04002000 |
| 0x00000009 | 0x20 |
| 0x0000000A | 0x00 |
| 0x0000000B | 0x04 |

Legend:

“bit” = one digit of binary (base-2)

“byte” = 8 bits

“word” = a group of bytes; how many depends on the computer’s architecture; let’s assume this computer has 4 bytes per word

0x45 = 4516

0x12345678 = 1234567816

1. How many bits in a byte?

8 bits

1. How many bytes in a word? How many bits in a word?

4 bytes per word; 32 bits in a word.

1. What is the address of the byte whose contents is 0x99?

0x00000004

1. What is the contents of the byte at address 0x00000003?

0x1A

1. How many unique addresses are there per byte of memory?

One (1)

1. How many addresses point within a given word of memory?

Four (4)

# Read This!

By convention, a word of memory is referred to by its *first* address.

1. What is the address of the word whose contents is 0xBF004199?

0x00000004 or 5 or 6 or 7

1. Write in the contents of the word of memory whose address is 0x00000000.

0x1A031500

# Model 2: Addresses and pointers

Here is a view of memory that is organized by word. Each cell of the memory contents is 1 byte. The addresses of individual bytes are not shown in the diagram.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| word address | memory contents | | | |
| 0x00300000 (y) | 0x00 | 0x30 | 0x00 | 0x14 |
| 0x00300004 |  |  |  |  |
| 0x00300008 |  |  |  |  |
| 0x0030000C |  |  |  |  |
| 0x00300010 |  |  |  |  |
| 0x00300014 (x) | 0x00 | 0x00 | 0x02 | 0x14 |
| 0x00300018 |  |  |  |  |
| 0x0030001C |  |  |  |  |
| 0x00300020 |  |  |  |  |
| 0x00300024 |  |  |  |  |
| 0x00300028 |  |  |  |  |

Assume sizeof(int) is 4; and sizeof(int\*) is 4.

Suppose this program ran and produced the memory diagram above.

int x = 532;

int \*y = &x;

1. What is the value of the word at memory address 0x00300000? at 0x00300014?

0x00300014, 0x00000214

1. What type of data is stored in variable x? (integer or address) How do you know?

Integer; because it =532.

1. What type of data is stored in variable y? (integer or address) How do you know?

It is an address because it has the address of another.

1. Where in memory is variable x stored? Variable y?

X is stored at 0x00300014, y is stored at 0x00300000.

1. Summarize why the memory contents looks the way it does, based on the code.

**Share!** Write your team’s answers to #9-12 on the board. You will present #13.

# Exercises

1. Draw on the memory diagram the result of this program…

int a = 45; // assume a lives at address 0x00000004

int \*b = &a; // assume b lives at address 0x00000010

int \*\*c = &b; // assume c lives at address 0x00000008

# Model 3: Arrays and memory

The program creates these values in memory

int arr[3];

arr[0] = 300;

arr[1] = 400;

arr[2] = 1024;

// assume sizeof(int) is 4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| word address | memory contents | | | |
| 0x00000000 |  |  |  |  |
| 0x00000004 |  |  |  |  |
| 0x00000008 |  |  |  |  |
| 0x0000000C |  |  |  |  |
| 0x00000010 |  |  |  |  |
| 0x00000014 | 0x00 | 0x00 | 0x01 | 0x2C |
| 0x00000018 | 0x00 | 0x00 | 0x01 | 0x90 |
| 0x0000001C | 0x00 | 0x00 | 0x04 | 0x00 |
| 0x00000020 |  |  |  |  |
| 0x00000024 |  |  |  |  |

1. Fill in the three blank columns.

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Size in bytes** | **Size in words** | **Starting address of this data** |
| arr[0] | 4 | 1 | 0x00000014 |
| arr[1] | 4 | 1 | 0x00000018 |
| arr[2] | 4 | 1 | 0x0000001C |
| the whole array | 12 | 3 | 0x00000014 |

1. Summarize how arrays are stored in memory.

If each word takes up 4 bytes, then each data point is going to be 4 bytes away from another data point in an array in terms of the memory address.

1. Using your table, come up with an equation relating the starting address of arr[i] to the index i.

Memory address[i] = starting address + (index i \* 4)

&arr[i] = &arr[0] + 4i

**Share!** Write your team’s answer to #17 on the board.

1. How did the size of an int affect your equation in #17?

Size of int affects how much you multiply i by.

1. Generalize your equation in #17, replacing the size of an int with a variable S.

Memory address[i] = starting address + i\*S

&arr[i] = &arr[0] + S\*i

# Read This!

To calculate the ***address of an element of an array***, you will always use the equation you came up with in #5. This calculation will be important when you write assembly code that deals with arrays.

# Exercises

Consider this program:

long arr[4]; // assume sizeof(long) is 8; sizeof(long\*) is 4

arr[0] = 10;

arr[1] = 11;

arr[2] = 12;

arr[3] = 13;

long \*z = arr;

1. Draw a fragment of memory after this program completes, assuming z lives at address 0x00000000, and the array starts at address 0x00000004.