Lab 6

Representation: A Hex Dump Program In C

Purpose

To learn how values in memory can be presented in hexadecimal form.

Background Reading And Preparation

Read Chapter 3 on data representation, and find both the integer and address sizes for the computer you use†. Ask the lab instructor for an exact specification for the output format.

Overview

Write a C procedure that produces a hexadecimal dump of memory in ASCII. The lab instructor will give details about the format for a particular computer, but the general form is as follows:

```
 Address         Words In Hexadecimal          ASCII characters
--------- -------- -------- -------- -------- ----------------
aaaaaaaa  xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx cccccccccccccccc
```

In the example, each line corresponds to a set of memory locations. The string *aaaaaaaa* denotes the starting memory address (in hexadecimal) for values on the line, *xxxxxxxx* denotes the value of a word in memory (also in hexadecimal), and *cccccccccccccccc* denotes the same memory locations when interpreted as ASCII characters. Note: the ASCII output only displays printable characters; all other characters are displayed as blanks.

Procedure And Details (checkmark as each is completed)

1. Create a procedure, *mdump* that takes two arguments that each specify an address in memory. The first argument specifies the address where the dump should start, and the second argument specifies the highest address that needs to be included in the dump. Test to ensure that the starting address is less than the ending address.

†On most computers, the address size equals the integer size.

2. Modify both arguments so each value specifies an appropriate word address (i.e., an exact multiple of four bytes). For the starting address, round down to the nearest word address; for the ending address, round up.

3. Test the procedure to verify that the addresses are rounded correctly.

4. Add code that uses *printf* to produce headings for the hexadecimal dump, and verify that the headings are correct.

5. Add code that iterates though the addresses and produces lines of hexadecimal values.

6. To verify that procedure *mdump* outputs correct values, declare a *struct* in memory, place values in fields, and invoke the procedure to format the values.

7. Add code that produces printable ASCII character values for each of the memory locations, as shown above.

8. Verify that only printable characters are included in the output (i.e., verify that a non-printable character such as 0x01 is mapped into a blank).

Optional Extensions (checkmark as each is completed)

9. Extend the dump program to start and stop on a byte address (i.e., omit leading values on the first line of output and trailing values on the last line).

10. Change the program to print values in decimal instead of ASCII character form.

11. Modify the dump program so instead of printing ASCII values, the program assumes the memory corresponds to machine instructions and gives mnemonic opcodes for each instruction. For example, if the first word on the line corresponds to a *load* instruction, print *load*.

12. Add an argument to procedure *mdump* that selects from among the various forms of output (ASCII characters, decimal, or instructions).