27.08.2024

Laboratório 4

Theo Canuto - 2311293

Professor Raúl Renteria

INF1018 - 3WA

2024.2

1. dump de c:
   endereço de p, conteúdo de p (em hexa)
   dump de s:
   endereço de s, conteúdo de s (em hexa) - 2 vezes por ser short (2 bytes, byte a byte)
   dump de i:
   endereço de i, conteúdo de i (em hexa) - 4 vezes por ser int (4 bytes, byte a byte)

2. Os valores mostrados são o endereço do byte e o valor daquele byte em hexadecimal, byte a byte no loop. O loop é separado byte a byte quando, antes dele se iniciar, é declarado um ponteiro para unsigned char que recebe o endereço do short l em uma chamada da função e do unsigned short k na outra. A última linha do loop passa para o próximo endereço, tornando possível essa lógica de percorrer byte a byte.

3.
```c
int xbyte (packed_t word, int bytenum) {
  int desloc = bytenum*8;
  word = word >> desloc;
  word = word & 0xff;
  if (word & 0x80) {
    word = word | 0xffffff00;
  } else {
    word = word & 0x000000ff;
  }
  return word;
}
```

4. O programa 1 compara o inteiro x (atribuído em hexa) com o inteiro y (atribuído em decimal). 0xffffffff é convertido para -1 pela lógica do complemento a dois, já que é um inteiro com sinal (por padrão, quando não é declarado unsigned, é signed).

   O programa 2 compara o inteiro unsigned x (atribuído em hexa) com o inteiro unsigned y (atribuído em decimal). Nesse caso, por ser unsigned, x vale 4294967295. O programa parece estar “errado” porque o x é printado como signed int (ou seja, -1), quando esse não é seu valor, pois foi declarado como unsigned.

   Por fim, o programa 3 compara o inteiro x (atribuído em hexa) com o inteiro unsigned y (atribuído em decimal). Nesse caso, x vale -1, por ser signed, e y segue valendo 2, independentemente da forma como está sendo armazenado, por ser um número menor que 2^8 bits.

5. No código gerado, a variável ui capturará o valor de sc, ou seja, sua representação interna em binário, e preencherá à esquerda os 24 bits que faltam para completar o tipo unsigned int (32-8=24). O valor decimal de sc é equivalente a -1. Utilizando a técnica de complemento a dois, subtraímos 1 de 1, resultando em 0, que corresponde a 0000 0000 em binário. Ao inverter os bits, obtemos 1111 1111. Ao chamar a função dump, podemos observar que a conversão para unsigned int preenche os bits à esquerda com 1s. Isso não se aplica apenas a -1, mas também a outros valores negativos do mesmo tipo.