1.6 Exercícios

1.1 Quantos bits são necessários para representar instruções em código de máquina para os seguintes processadores? Assuma que todos os códigos de operação têm o mesmo número de bits.

(a) Um processador com 38 instruções que podem ter referências a dois endereços de memória de 32 bits cada um;

(b) Um processador com 32 instruções que podem ter referências a três registradores, sendo que há 16 registradores no processador;

(c) Um processador com 142 instruções que podem ter referências a um endereço de 32 bits.

1.2 Um dos projetistas do processador hipotético da Seção 1.1.1 sugeriu que, em vez das instruções `LOAD` e `STORE`, uma única instrução `MOVE` deveria ser utilizada; nesse caso, a ordem dos operandos serviria para diferenciar a direção da transferência. Por exemplo, `MOVE 1,R0` indicaria a transferência do conteúdo da posição 1 de memória para o registrador R0. Já a instrução `MOVE R0,4` indicaria a transferência do conteúdo do registrador R0 para a posição de memória 4. Por que os projetistas teriam mantido duas instruções separadas?

1.3 Qual é o código binário para as seguintes instruções do processador hipotético da Seção 1.1.1?

(a) `LOAD 1,R0`

(b) `STORE R0,4`

(c) `BZERO R2,15`

(d) `ADD R0,R2,R0`

1.4 Qual é a seqüência de instruções simbólicas correspondentes ao seguintes códigos de máquina do processador hipotético da Seção 1.1.1?

(a) `00000000 10001010 11101010`

(b) `11111111 10101010 01010101 00011011`

1.5 Os projetistas da segunda geração do processador hipotético da Seção 1.1.1 devem considerar as seguintes demandas:

(a) Acrescentar duas novas operações;

(b) Dobrar o número de registradores de dados;

(c) Dobrar a largura dos dados de 8 para 16 bits;

(d) Ampliar a capacidade de endereçamento de 16 para 256 posições.

Qual o impacto isolado de cada uma dessas modificações no formato binário das instruções do processador? E, se todas as modificações forem implantadas, qual será o novo formato da instrução?

1.6 Um programa em C ou C++ permite a passagem de argumentos da linha de comando por meio de dois parâmetros da função `main`:

```
int main(int argc, char *argv[]) {
    ...
}
```

O primeiro parâmetro, que tipicamente recebe o nome `argc` (*argument count*), indica o número de palavras (separadas por espaços) presentes na linha de comando, incluindo o próprio nome do programa. O segundo parâmetro, cujo nome típico é `argv` (*argument value*), é um arranjo de ponteiros para caracteres, onde cada elemento do arranjo representa uma das palavras da linha de comando. Com o uso desses argumentos, desenvolva um programa em C++ para apresentar na saída padrão o conteúdo de um arquivo cujo nome é fornecido na linha de comando.

1.7 Com o uso de `argc` e `argv`, definidos anteriormente, desenvolva um programa em C++ para implementar a cópia do conteúdo de um arquivo, cujo nome é passado como o primeiro argumento para o programa na linha de comando, para outro arquivo, cujo nome é passado como o segundo argumento na linha de comando.

1.8 Com o uso de `argc` e `argv`, definidos anteriormente, desenvolva um programa em C++ para contar o número de caracteres, palavras e linhas no arquivo cujo nome foi especificado na linha de comando e apresentar esses totais na tela (saída padrão).

1.9 Qual é o erro associado a cada uma das seguintes declarações de variáveis em um programa C++? Com o auxílio de um compilador C++, interprete as mensagens associadas a esses erros.

(a) int do;

(b) int valor = 078;

(c) char a.c = 0;

(d) char b = 715.

1.10 A função `atoi`, da biblioteca padrão da linguagem C, permite a conversão de uma seqüência de caracteres (passada como argumento da função) para um valor inteiro (seu valor de retorno). Use essa função para implementar

uma função C++ que receba qualquer quantidade de inteiros na linha de comando e apresente na saída padrão a soma desses valores. Por exemplo, se o programa executável tem o nome de `total`, a execução

```
total 1 20 100
```

deve apresentar na tela o valor 121.

1.11 A partir do exemplo da Seção 1.4.2, apresente três outras situações de má formação de arquivos XML.

1.12 Por que, no segundo exemplo de erro no processamento do arquivo XML apresentado na Seção 1.4.2, a indicação de ausência da marcação de fim do elemento `livro` só apareceu na última linha do arquivo, quando o analisador encontrou a marcação de final do elemento raiz, e não quando o segundo elemento `livro` foi iniciado?