12º Exercício Prático – parte 1
Desenvolvido no Laboratório

Objetivo

Observar como aplicar as instruções SIND disponíveis no processador da arquitetura IA-32 e Intel 64.

Materiais

1. Compilador GCC
2. Arquivo Acessos.cpp
3. Arquivo com textos: biblia-em-txt.txt.

Introdução

O experimento de hoje usa novamente um programa que faz a contagem de ocorrências de uma letra em um arquivo de texto. Os processadores da Intel que equipam os computadores pessoais oferecem a possibilidade de executar programas que usam também instruções que executam paralelamente uma instrução em vários dados de uma única vez (SIMD).

Desenvolvimento

A intenção é comparar a otimização -O2 com a -O3 que usa instruções SIMD e apresentar um programa escrito em linguagem de montagem que também usa instruções SIMD.

Executando o programa

1. Compile o programa Acessos.cpp
   - gcc -masm=intel -g -O2 Acessos.cpp -o Acessos
2. Execute o programa.
   - ./Acessos

Qual foi o tempo observado na execução?

Tempo 1: _____ s

Usando a otimização do compilador -O3

A opção -O3 ativa no compilador um nível de otimização ainda maior que o -O2, muitas destas otimizações usam instruções SIMD. Existem boatos que este nível de otimização nem sempre gera códigos confiáveis.

1. Compile o programa AcessosA .c
   - gcc -masm=intel -g **-O3** Acessos.cpp -o Acessos
2. Execute o programa.
   - ./Acessos

Qual foi o tempo observado na execução?

Tempo 2: _____ s

Usando linguagem de montagem e instruções SIMD

As instruções SIMD podem ser usadas para construir rotinas mais otimizadas que executam uma instrução em muitos dados de uma só vez.

1. Abra o arquivo Acessos.cpp em um editor de texto
2. Altere a função contaLetra() como código abaixo

```
int contaLetra(FILE *pArq, char c) {
  int cont = 0;
  int tam = fread(tudo, MAX_CARACTERES_NA_BIBLIA, 1, pArq);
  char carac[16] = {c, c, c, c, c, c, c, c, c, c, c, c, c, c, c, c};
  asm("   MOV       RDI,  %[tudo]     \n"
      "   MOV       RCX,  %[MAX_CARACTERES_NA_BIBLIA] \n"
      "   MOVAPS    XMM1, %[carac]    \n" // carrega XMM1 com c repetido 16 vezes
      "   XOR       RBX,  RBX         \n" // limpa RBX
      "   ADD       RCX,  RDI         \n" // calcula endereço final
      "repete%=:                      \n"
      "   MOVAPS    XMM0, [RDI]       \n" // carrega XMM0 com 16 letras do texto
      "   PCMPEQB   XMM0, XMM1        \n" // compara byte a byte XMM0 com XMM1
      "   PMOVMSKB  RAX,  XMM0        \n" // junta os MSB de cada byte de XMM0 em RAX
      "   POPCNT    RAX,  RAX         \n" // contas quantas letras 'a's foram encontradas
      "   ADD       RBX,  RAX         \n" // soma o numero de letras 'a's
      "   ADD       RDI,  16          \n" // salta para pegar as próximas 16 letras
      "   CMP       RDI,  RCX         \n" // compara se chegou ao final do texto
      "   JL        repete%=          \n" // repete até chegar no final
      "   MOV       %[cont], EBX      \n" // retorna a contagem
      : [cont] "=r"(cont) // resultado em cont
      : [tudo] "r"(tudo), // vetor com texto
        [MAX_CARACTERES_NA_BIBLIA] "i"(MAX_CARACTERES_NA_BIBLIA),
        [carac] "m"(carac)
      : "rax", "rbx", "rcx", "rdi", "xmm0", "xmm1");
  return cont;
}
```

Qual foi o tempo observado na execução?

Tempo 3: _____ s

Avaliando os resultados

Envie a avaliação dos resultados como descrito no arquivo “Avaliacao Dos Resultados.pdf”.

Conclusão

As instruções SIMD pode trazer alguma vantagem no tempo de execução de um programa quando bem utilizadas.