Guião Laboratorial 3

Loop-Unrolling e Processamento vetorial

Objetivos:

- introdução ao *loop-unrolling*
- exploração das instruções vetoriais para processamento paralelo

Introdução:

A técnica de *loop-unrolling* consiste em fazer várias cópias do núcleo de um ciclo (*loop*).
Um exemplo de *loop-unrolling* de grau 2 é apresentado na tabela abaixo. Os atuais compiladores implementam o desdobramento do ciclo de forma automática (no `gcc` é necessário ativar a otimização), sem necessidade de implementação por parte do programador.

no unroll	unroll-2

`...`
`para k= 0 até N-1, k++`
`  C[i][j] += A[i][k] * B[k][j]`
`...`
	`...`
	`para k= 0 até N-1, k+=2`
	`  C[i][j] += A[i][k] * B[k][j]`
	`  C[i][j] += A[i][k+1] * B[k+1][j]`
	`...`

Este guião está divido 5 exercícios. Os 4 primeiros exercícios devem ser resolvidos durante a aula prática. O exercício 5 é para os alunos resolverem em casa para preparação da próxima aula pratica. Para a elaboração deste guião são necessários os ficheiros da aula anterior.

Exercício 1- *Loop-unroling*

a) Apresente quais as vantagens de utilização da técnica?

b) Utilize a opção `-funroll-loops` no `gcc` para otimizar a função `gemm1()`. Analise o código *assembly* gerado. Identifique quantas vezes foi desdobrado o ciclo e estime o total de instruções executadas (utilize o tamanho 512). Apresente uma estimativa para o ganho obtido com esta otimização (compare a versão base com a versão otimizada).

Exercício 2 - Medição do impacto do *loop-unrolling*

a) Execute novamente a versão `gemm1()` com a otimização de *loop-unrolling* (`srun -partition=cpar ./gemm 512 1`) e compare com os resultados obtidos em aulas anteriores. Preencha a Tabela 2 e comente as diferenças quanto ao número de instruções? Calcule o ganho obtido quanto número de ciclos executados.

n	#CC	CPI	#I
original			
loop-unrolling estimado			
loop-unrolling			

Tabela 1 - Impacto do *loop-unrolling*

Exercício 3 - Análise da versão ijk (dot)

a) O compilador não consegue vetorizar automaticamente a versão dot (ijk). Identifique os motivos através da análise do padrão de acessos às matrizes A,B e C (localidade espacial, escritas concorrentes).

b) É possível trocar a ordem de execução dos ciclos originando diferentes desempenhos. Qual ou quais as variantes que facilitam a geração de instruções vetoriais?

Exercício 4 - Utilização de instruções vetoriais (análise + medição)

Altere os ficheiros da aula anterior:

1. Em `gemm.c` copie o código de `gemm1()` para `gemm4()`, e altere o algoritmo de forma a implementar a versão que escolheu no exercício anterior.
2. Altere a assinatura da função de modo a informar o compilador que os apontadores apontam para zonas de memória distinta, para tal, quantifique os apontadores `a`, `b` e `c` como `__restrict__`.
3. Adicione a opção `-ftree-vectorize, -msse4` à compilação para ativar a vectorização com instruções SSE4.
4. Remova a opção `-funroll-loops` da compilação.

a) Quantos elementos são calculados de cada vez? Estime a quantidade de instruções executadas, preencha a respetiva coluna na Tabela 2.

Nota: Assuma que a variante identificada anteriormente utiliza instruções vetoriais no cálculo e que é executada numa máquina com vetores de 128bits (exemplo Intel SSE4).

b) Execute a versão `gemm4()` (`srun --partition=cpar ./gemm 32 4`) da multiplicação de matrizes e preencha a Tabela 2. Como aumentam as várias grandezas em função do tamanho da matriz? Discuta as diferenças entre o ganho esperado e o ganho medido?

n	#CC	CPI	#I	#I estimados
32				
128				
512				

Tabela 2 - Impacto do processamento vetorial(kij)

Exercício 5 - OpenMP para Kernel ijk

Utilize o OpenMP para forçar o compilador a gerar uma versão com instruções vetoriais com a ordem dos ciclos ijk.

Utilize os ficheiros da aula anterior:

1. Em `gemm.c` copie o código de gemm1 para gemm4.
2. Altere o código de forma a assumir que a função `gemm5` é passada a matriz B^T.
3. Utilize a primitiva `#pragma omp simd reduction(+:cij)` no ciclo interior para forçar a geração de instruções vetoriais.
4. Altere a `Makefile` de modo a suportar o `OpenMP`, adicione a flag `-fopenmp`.
5. Use uma versão do gcc que suporte `OpenMP4`, `module load gcc/5.3.0 3.`

a) Execute a versão `gemm4()` (`srun --partition=cpar ./gemm 32 5`) da multiplicação de matrizes e preencha a Tabela 3. O que justifica a diminuição do tempo de execução (#CC mais baixo) em relação a versão ikj (análise o kernel)?

n	#CC	CPI	#I
32			
128			
512			

Tabela 3 - Impacto do processamento vetorial(ijk)