Curso: LEF/MEI/MIEF/MIEI **Teste modelo**
U.C.: Computação/Programação Paralela **Duração:** 2h30m

1. [15]A especificação da SVE na arquitetura ARM suporta execução de operações com vetores de 32 *doubles* (precisão dupla). Contudo, a Fujitsu no desenho do seu novo componente de 48 cores optou por construir a unidade vetorial para operar com vetores de apenas 8 *doubles*. **Apresente** uma explicação para esta opção.

2. [30]Na arquitetura das unidades de processamento mais recentes, quer como extensões de CPU convencionais, quer como novos dispositivos (GPU, TPU, ...) tem-se vindo a assistir ao aparecimento de operações com tensores.

   a) [10]**Caracterize** sumariamente estas operações e **indique/explique** a sua relevância atual em sistemas de computação.

   b) [10]A comunidade científica sempre foi muito exigente quanto à precisão dos números reais usados em computação, exigindo normalmente valores de precisão dupla ou superior. Contudo, a mesma comunidade científica, com o aparecimento destas novas arquiteturas, começou a pedir agora o inverso: representação de reais com "meia-precisão", preferindo até formatos não *standard* (como o seguido pela Google, o *bfloat*), ou mesmo usando tensores representados por 8 ou 4 bits. **Apresente** uma explicação para esta inversão de valores da mesma comunidade.

   c) [10]A Intel apostou aqui em 2 frentes distintas: por um lado na extensão AMX, por outro em novos componentes específicos para redes neuronais. **Caracterize** estas duas abordagens.

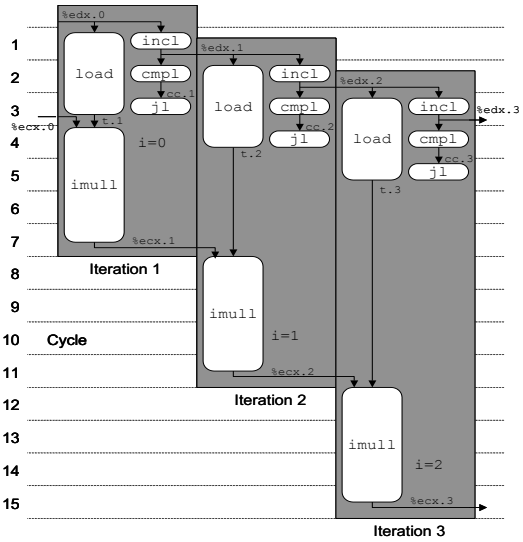
3. [15]A Intel desistiu de dar continuidade à linha Xeon Phi (*manycore*), mas decidiu aproveitar alguns dos aspetos positivos dessa linha para a introduzir na família do *Scalable Processors*, ao mesmo tempo que removeu alguns dos aspetos negativos.
   Identifique e **caracterize** sumariamente estes aspetos positivos e os negativos

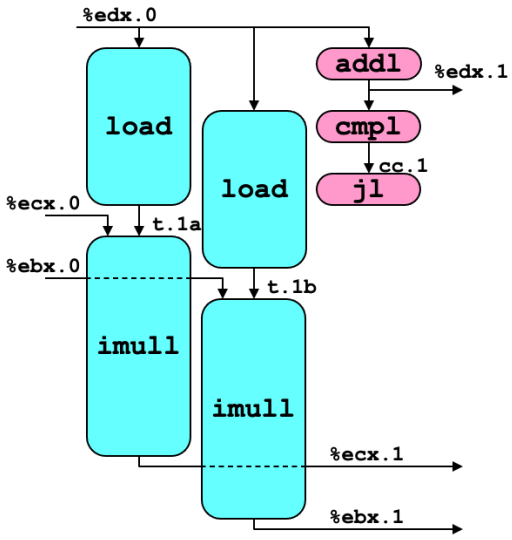
4. [20]Considere esta figura que mostra os 4 melhores sistemas HPC na lista de TOP500 de junho de 2020, em especial os sistemas colocados em 1º e em 4º lugar. **Explique** como o sistema chinês, com maior nº de *cores* (e em que cada *CPU-chip* tem 256+4 *cores*), tenha um valor medido de desempenho máximo mais de 4 vezes inferior ao sistema japonês.

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371

5. [20]O arquitetura do sistema Deucalion que foi adquirido para o MACC é constituído por servidores similares aos sistemas que se encontram em #1 e #5 no TOP500 de Nov-20. **Caracterize** sumariamente as arquiteturas desses servidores (apenas em termos de unidades de processamento/aceleração e memória primária).

6. [30]Considere as seguintes figuras que ilustram possíveis execuções de um ciclo `for` para multiplicação de todos os elementos dum vetor (tipo `int`), num P6 a 1GHz.
O corpo do ciclo `for` na fig da esquerda é apenas constituído pela instrução `x *= data[i]`, enquanto a fig da direita mostra a introdução de alterações ao código C para melhorar o desempenho.





a) [15]**Indique**, **justificando**, <u>todas as características</u> que deveria ter este processador P6 para que pudesse executar o código da função com este desempenho de 4 CPE.

b) [15]**Identifique** as alterações introduzidas no código para que a sua execução pudesse ser representada pela fig da direita. **Estime** o ganho teórico de desempenho que seria de esperar com estas alterações ao código, face ao valor anterior de 4 CPE.
(Sugestão: desenhando o fluxo de execução de 3 iterações do ciclo torna-se mais fácil...)

7. [30]Considere a seguinte função para ordenar um vetor de inteiros, inspirada no "*radix sort*".

```
typedef struct {
         int tam;
         int *vals;
 } digitos;

void rsort(int v[],int tam);   // função externa
int get_MSDigit(int val);      // função externa
// valor entre 0..15 (digito mais significativo)

void radix_sort(int v[],int tam){
  int k=0;
  digitos dig[16];

  // iniciar a estrutura dig
  for (int i=0; i<16; i++) {
      dig[i].tam=0;
      dig[i].vals = malloc(tam);
   }

  // colocar os valores na estrutura dig
  for (int i=0; i<tam; i++) {
      int d = get_MSDigit(v[i]);
      dig[d].vals[ dig[d].tam++ ]=v[i];
  }

  // ordenar cada elemento da estrutura dig
  for (int i=0;i<16;i++) {
      rsort(dig[i].vals, dig[i].tam);
  }

  // colocar valores no vetor original
  for (int i=0; i<16; i++) {
      for (int j=0; j<dig[i].tam; j++)
          v[k++] = dig[i].vals[j];
  }
}
```

a) [15]Questões relativas à análise e otimização deste algoritmo:

i. Indique a complexidade de cada uma das fases de execução do algoritmo.

ii. Indique, **justificando**, as fases de execução do algoritmo que devem ser otimizadas para melhorar o seu desempenho e apresente para cada uma delas uma possível otimização.

iii. Escolha uma das otimizações que apresentou anteriormente e mostre a sua implementação

b) [15]**Apresente** uma versão paralela desta função em **OpenMP** e **justifique** as opções que tomar para cada um dos 4 ciclos do programa.
(**Sugestão**: utilize o espaço disponível à direita da figura.)
Indique a partir de que #threads é que é esperado que a *performance* do algoritmo deixe de escalar.

8. [30]Desenvolva uma versão paralela do programa para execução em memória distribuída, com passagem de mensagens (i.e., MPI). Considere que o processo 0 contém todos os dados do problema no início da execução da função e que no final deverá conter o *array* ordenado.

a) [10]**Apresente** o padrão que utilizará para especificar a versão paralela em memória distribuída e **identifique** as trocas de mensagens necessárias entre os vários processos.
(**Sugestão**: mostre através de esquema para uma execução com 3 processos.)

b) [20]**Implemente** a versão paralela para execução em memória distribuída.

9. [20]**Desenvolva** uma implementação eficiente de um algoritmo que aplique uma máscara em CUDA. **(Sugestão**: complete o código apresentado em baixo)
Este algoritmo deverá aplicar a máscara *`mask` ao *array* *`in`, usando o operador *bitwise* `&`, guardando o resultado no *array* *`out`.
Considere que o tamanho dos *arrays* *`in` e *`out` é igual ao total de *threads* lançadas, e que o tamanho da máscara é variável.

```
__global__
void apply_mask (int *in, int *out, unsigned *mask, int mask_size) {
     unsigned tid = treadId.x + blockId.x * blockDim.x;
     // ...
}
```