Quiz 2: Solutions

EEE4084F

2015-03-19

Question 1: Processor Architectures **[15 Total]**

#1 All four of: **[4]**

- Single Instruction, Single Data (SISD)
- Single Instruction, Multiple Data (SIMD)
- Multiple Instruction, Single Data (MISD)
- Multiple Instruction, Multiple Data (MIMD)

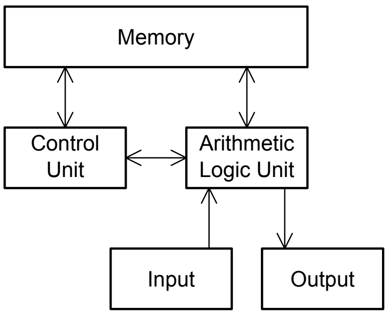
#2 Lockstep execution refers to a synchronised operation where each processing core starts and ends execution of the same instruction at the same point in time. **[2]**

#3 SIMD **[1]**

#4 Each SIMD core of a GPU generally operates in lockstep. **[1]**

#5 SISD **[1]**

#6 The figure below shows the classic Von Neumann architecture.



The memory stores both program instructions and data. The control unit decodes the instructions and tells the arithmetic logic unit (ALU) what to do. The ALU performs elementary operations, such as addition, subtraction and moving bytes of data between input, output, memory and internal registers. 'Input' and 'Output' refers to peripherals such as long term storage, keyboard, screen and printer. **[6]**

#7 The Harvard Architecture has separate data and program memory. Accessing program and data can therefore occur simultaneously and independently, which eliminates the memory sharing problems of the Von Neumann architecture.

The Von Neumann architecture has the advantage in that it is more flexible. It does not have fixed program and data memory sizes, so it can be used for a wider variety of applications: some program intensive and others data intensive. **[3 (bonus)]**

Question 2: Parallel Computing Design **[15 Total]**

#1 When referring to processing: Latency refers to the time taken for data to move through the entire processing chain; Bandwidth refers to the maximum possible data-rate through the processing chain (often limited to short bursts); Throughput refers to the real-world average data-rate through the processing chain.

When referring to memory: Latency refers to the time taken to fetch data from memory, including all overhead involved; Bandwidth refers to the maximum possible data-rate between the memory device and the target device (often limited to short bursts); Throughput refers to the real-word average data-rate between the memory device and the target device. **[5]**

#2 Functional decomposition focuses on the computation to be performed. The problem is decomposed into tasks according to the work that must be done. Each task performs a portion of the overall work.

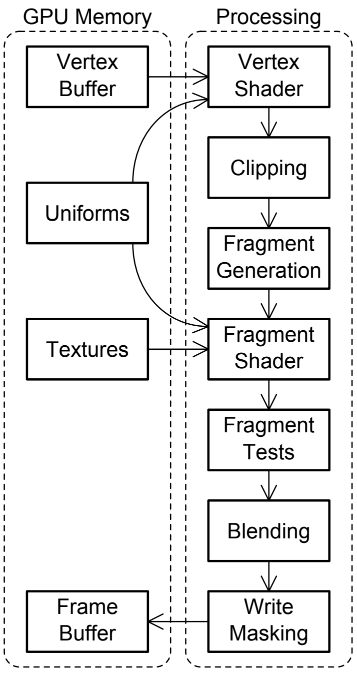
Domain decomposition focuses on the data that must be processed. The data is decomposed into smaller data-sets, or different views of the same data is processed by the same function. Each parallel task works on its own portion of the data, or does the same thing to different views of the same data. **[4]**

#3 Categorised as follows: **[6]**

- Convert an image from colour to grey-scale.
  Embarrassingly parallel – each pixel in the source image is processed completely independently.
- Apply a 5×5 pixel Gaussian filter to a 1 024×1 024 picture.
  Course-grained – each resulting pixel can be calculated independently, even though the source data is shared between neighbouring resulting pixels.
- Finite element simulation of a mechanical structure.
  Fine-grained – each iteration of the algorithm changes the structure shape (and other parameters), so there is heavy communication between the threads after every time-step.

Question 3: OpenGL **[10 Total]**

#1 Below is a brief overview of the graphics pipeline (more detail than the question requires). For 4 marks I expect at least the Vertex Shader, Fragment Generation (Rasteriser), Fragment Shader and Write-To-Frame Buffer steps of the pipeline. To obtain the final mark, explain briefly the concepts of vertices and uniforms. **[5]**



The figure on the right shows a simplified version of the OpenGL graphics pipeline. The vertex buffer is filled with vertex attribute data. Attributes can be anything the application requires, but typically include vertex spacial coordinates, colour and texture coordinates.

Uniforms are constant for all vertices and fragments. These typically include the transform matrix, texture handles, lighting positions, etc. Textures are large blocks of data that can be used in any way the application requires. In the simplest case, these blocks of memory are filled with pictures to be mapped onto the model being drawn.

The vertex shader is a program that runs for every vertex in the vertex buffer provided. It typically applies matrix transformation. The vertex shader has to output at least the spacial coordinates of the vertex within the so-called unit-cube. This cube is an abstract space that the GPU uses to clip fragments that fall outside the screen area. It includes the range [-1; 1] in the X, Y and Z dimensions. The vertex shader can also output any other attributes that the fragment shader requires. All these attributes are then linearly interpolated to generate fragment attributes. The resulting fragment attributes are then given to the fragment shader.

The fragment shader typically applies texture mapping, lighting effects, etc. It can either discard the fragment, or output the fragment colour to the next stage in the pipeline.

After all this processing, the frame buffer is updated with the new fragment colour.

#2 Any logically complete list of five of: **[5]**

- Create a window and then a rendering context for the window
- Compile and load the shader programs
- Copy the textures to the GPU memory (optional)
- Build and copy the vertex buffer(s) to the GPU memory
- Tell the GPU what the uniform values are
- Tell the GPU where which attribute is stored within the vertex buffer(s)
- Tell the GPU to run the currently bound vertex buffer
- Swap the frame buffers so that the previous back buffer is drawn to screen.