**CS570 Advance Computer Architecture**

**Homework 9**

1. **(12 points) Array processors, Vector processors, and VLIW architecture, all of them can achieve parallel execution. Please describe the difference between them. Please give the differences from both hardware (execution model) and software (program model) perspectives.**

* ***Array processors*** are also known as multiprocessors or vector processors. They perform computations on large arrays of data. Thus, they are used to improve the performance of the computer.
* In computer languages, **vectors** are a one-dimensional data structure. **Arrays**, in turn, can have more than one dimension. But “***vector processing***” refers to a specific way that microprocessors have to perform the same operation on several inputs at the same time (in one instruction).
* ***Very long instruction word (VLIW)*** describes a computer processing architecture in which a language [compiler](https://whatis.techtarget.com/definition/compiler) or pre-processor breaks program instruction down into basic operations that can be performed by the [processor](https://whatis.techtarget.com/definition/processor) in parallel (that is, at the same time).

**Execution Model and Program Model Differences:**

***Array processor***: Instruction operates on multiple data elements at the same time. Array processor is composed of index collections of data or information normally known as indices but having more then one index collection is not necessary

***Vector processor***: Instruction operates on multiple data elements in consecutive time steps. vector word used when there are two indices at least.

***VLIW***: You can execute completely different instructions in parallel i.e executes more than one instruction at a time a characteristic referred to as *superscalar*.

**2. (12 points) Use brief words to describe the difference of the following terms: SIMD, Data flow, Thread(control)-level Parallelism, SIMT, SPMD.**

**SIMD**

Single instruction operates on multiple data elements. No of processors running the same instruction one clock cycle by the strict lock approach and it exploits instruction-level parallelism.

**Data flow**

Concurrency arises from executing different operations in parallel (in a data driven manner).

**Thread(control)-level Parallelism**

Concurrency arises from executing different threads of control in parallel. Splitting program into independent tasks

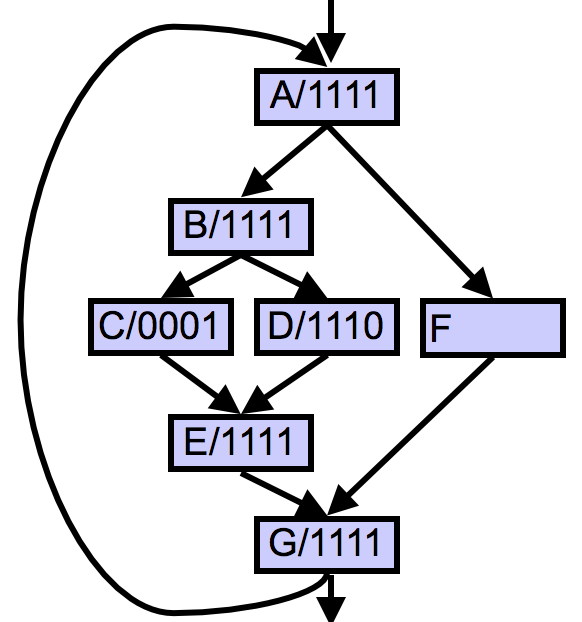
**SIMT**

Single instruction, multiple thread (SIMT) is an execution model used in [parallel computing](https://en.wikipedia.org/wiki/Parallel_computing) where [single instruction, multiple data](https://en.wikipedia.org/wiki/Single_instruction,_multiple_data) (SIMD) is combined with [multithreading](https://en.wikipedia.org/wiki/Thread_(computing)#Multithreading). The SIMT execution model has been implemented on several GPUs

**SPMD**

Single program multiple data, SPMD mode is a method of parallel computing its processors run the same program but execute the different data. SPMD could get the better performance through increasing the number of processors. Example of SPDM is MPI.

1. **(12 points) GPU can handle branch divergence by employing “SIMT-Stack” to record the PC and Active mask. Please read the related slides and text, and the following branch divergence flow. Then draw the “Stack Status” figure step by step when the instruction stream is from A to G.**



1. **(13 points) Solve 4.9 of the text (6th edition)**

**Consider the following code, which multiplies two vectors that contain single-precision complex values:**

**for (i=0;i < ;i++)**

**{  
c\_re[i] = a\_re[i] \* b\_re[i] - a\_im[i] \* b\_im[i];**

**c\_im[i] = a\_re[i] \* b\_im[i] + a\_im[i] \* b\_re[i];**

**}**

**Assume that the processor runs at 700 MHz and has a maximum vector length of 64. The load/store unit has a start-up overhead of 15 cycles; the multiply unit, 8 cycles; and the add/subtract unit, 5 cycles.**

**A close up of text on a white background

Description automatically generatedA close up of text on a white background

Description automatically generated**

1. **(13 points) Solve 4.13 of the text (6th edition) Assume a GPU architecture that contains 10 SIMD processors. Each SIMD instruction has a width of 32 and each SIMD processor contains 8 lanes for single-precision arithmetic and load/store instructions, meaning that each non diverged SIMD instruction can produce 32 results every 4 cycles. Assume a kernel that has divergent branches that causes, on average, 80% of threads to be active. Assume that 70% of all SIMD instructions executed are single-precision arithmetic and 20% are load/store. Because not all memory latencies are covered, assume an average SIMD instruction issue rate of 0.85. Assume that the GPU has a clock speed of 1.5 GHz.**

**A close up of a piece of paper

Description automatically generated**

1. **(13 points) Solve 4.15 of the text (6th edition)**

**List and describe at least four factors that influence the performance of GPU kernels. In other words, which runtime behaviors that are caused by the kernel code cause a reduction in resource utilization during kernel execution?**

Typically, the performance of a GPU kernel is limited by one of the following factors:

* memory throughput
* Instruction throughput
* Latency
* a combination of the above.

There exist three ways to determine which of those limits the performance of your kernel.

1. **(13 points) Solve 4.16 of the text (6th edition)**

**Assume a hypothetical GPU with the following characteristics: Clock rate 1.5 GHz ,Contains 16 SIMD processors, each containing 16 single-precision floating- point units Has 100 GB/s off-chip memory bandwidth**

**Without considering memory bandwidth, what is the peak single-precision floating-point throughput for this GPU in GLFOP/s, assuming that all memory latencies can be hidden? Is this throughput sustainable given the memory band- width limitation?**

**A close up of text on a white background

Description automatically generated**

**8. (12 points) As Amdahl’s law reveals, the speedup has an upper bound of 1/(1-f). This reminds us the early days of supercomputers. At that time, all major manufactures made supercomputers with up to 8 processors, citing Amdahl’s law. However, nowadays, we have multicore processors with much more cores (Intel MIC 64 cores) on the same die.**

**a) What is the limitation of Amdahl’s law? Is the “fixed-time” and “memory bounded” speedup model more appropriate for modern multicore design? Please answer the questions and tell the reason.**

* Amdahl's Law does clearly state a limitation of parallel computing. But this limitation varies not only from problem to problem, but with the size of the problem as well.
* Amdahl’s “fixed-size speedup” had a limited view on the potential of parallelism, Memory bounded modeling demonstrates that the memory size is a limiting factor for parallel scalability, hence memory bounded speedup model is more appropriate for modern multicore design

**b) What is the influence of data access in the scalability of multicore design, in terms of the number of cores?**

The contributions of three main speedup models (fixed-size, fixed-time, and memory-bounded speedups models) to the multi-core era, presents a very optimistic view. However, their view assumes that the data-access delay is fixed and independent of the number of cores and problem sizes. This assumption is often unrealistic because of the memory wall, caused by the increasing data-access delay as the number of cores increases.

Increasing the number of cores can have an even more significant effect on the data-access delay depending on the architecture’s characteristics.