Midterm Exam
2 Hours: Closed Book
El-Ghazawi

Be concise and brief. The space allocated is an indication of how concise your answer should be.

1. [a] Match each of the members of list A with the most related one in list B.

List A:
1 - Non Uniform Memory Access (NUMA);
2 - Von Neumann Machine;
3 - Data Flow Architectures;
4 - No Remote Memory Access (NORMA);
5 - Single Instruction Multiple Data (SIMD);
6 - Single Program Multiple Data (SPMD)

List B:
3-An architecture in which processing starts when input data are ready;
2-An architecture in which program and data are stored in the main memory;
1-Distributed Shared Memory Architecture;
6-A mode/method to emulate SIMD on MIMD machines;
4-a Multicomputer;
5-A parallel architecture in which synchronization is done at the hardware level.

[b] Describe two unique features in each of the following parallel architectures

i. Parallel Vector Processor (PVP)
   1. PVP systems contain a small number of powerful custom-designed vector processors (VPs) to manipulate vector data
   2. PVP systems normally do not use caches but they use a large number of vector registers and an instruction buffer

ii. Symmetric Multiprocessor (SMP)
   1. Every processor has equal access to the shared memory, the I/O device, and the operating system devices.
   2. In SMPs Processors are connected to shared memory through a bus or crossbar system interconnect, that are both difficult to scale once built

iii. Massively Parallel Processor (MPP)
   1. A very large scale computer system with commodity processing nodes interconnected together with a high speed low latency interconnect. Memories are physically distributed
   2. Only a few hosts OS have a micro-Kernel
iv. Cluster of Workstations (COW)
1 Each node is itself a complete workstation, without some peripherals
2 There is a complete OS residing on each node. The OS of a COW is the same as a UNIX workstation, plus an add-on software layer to support single system image, availability, parallelism, communication, and/or load balancing

v. Distributed Shared Memory (DSM) Machine
1 The memory is physical distributed among different nodes, but system hardware and software create an illusion of a single address space to application layer
2 Special hardware and software extensions are used to maintain memory consistency and coherence
2. [a] Parallel execution of applications involves additional activities that go beyond the basic computations in the application, due to parallel processing. Such activities are referred to as Parallel Overhead. List and define very concisely the three most important forms of parallel overhead.

i. Interprocessor Communications
ii. Synchronization
iii. Imbalance
iv. Parallel Redundancy

[b] Formulate the Amdhal’s Law and state its implications for parallel processing

\[
S = \frac{1}{f + \frac{1}{n}(1 - f)} , \quad f \text{ is the sequential fraction}
\]

[c] Sketch the speed up and efficiency for a parallel application in which 20% of the code is inherently sequential

![Graph showing speed up and efficiency](image)

[d] Unlike the initial belief, Amdahl’s law did not turn out to be an obstacle for parallel computing. Explain?

Gustafson has shown that as the problem size increases, the serial fraction decreases. Thus, parallel processing can help large problems.
3. A 500 MHz processor has a linear instruction pipeline constructed with 5 stages for fetch, decode, read operand, execute, and write result.

a. What would be the exact throughput (in MIPS), efficiency, and speed up over a non-pipelined version if 80 instructions were executed?

b. Repeat a if the pipeline was converted to a 2-issue superscalar and a superpipeline driven by a clock which is 4 times faster

c. Repeat a if the probability of an instruction to be a successful branch is 0.2

- Pipelined Time = k + n -1
  = 5+ 80 -1 = 84 clk

  non Pipelined time = 80*5 = 400 clk

  \[ t = \frac{1}{500*10^6} = 2 * 10^{-9} = \text{2 n. sec} \]

  Throughput = \( \frac{80}{84 * 2 * 10^{-9}} \) = \( 0.476 * 10^9 \) ins/sec = 476 MIPs

  Speed up = \( \frac{400}{84} = 4.76 \)

  Efficiency = \( \frac{4.76}{5} = 0.952 \)

b- With superscalar superpipeline

  Pipelined Time = k + (\( \lceil n/2 \rceil -1 \))*(1/4)
  = 5+ (40 -1) * 1/4 = 14.75 clk

  Throughput = \( \frac{80}{14.75 * 2 * 10^{-9}} \) = \( 2.711 * 10^9 \) ins/sec = 2711 MIPs

  Speed up = \( \frac{400}{14.75} = 27.11 \)

  Efficiency = \( \frac{27.11}{5*2*4} = 0.677 \)

c- With branch

  Pipelined Time = k + n -1+ pq n (k-1)
  = 5+ 80 -1 + 0.2 * 80 *4 = 84 +64 = 148 clk

  Throughput = \( \frac{80}{148 * 2 * 10^{-9}} \) = \( 0.270 * 10^9 \) ins/sec = 270 MIPs

  Speed up = \( \frac{400}{148} = 2.7 \)

  Efficiency = \( \frac{2.7}{5} = 0.54 \)
4. 
   a) List two unique features for each of the following programming models and give one example of a parallel language or library implementation of this model: Message Passing, Data Parallel, Shared Memory, and Distributed Shared Memory.
   
   **Message Passing:** Programmers control data and work distribution  
   - Explicit send and receive (Two sides)  
   - Example: MPI

   **Data Parallel:** Easy to write and comprehend  
   - No synchronization required (built in)  
   - Example: HPF

   **Shared Memory:** Simple statements: read remote memory via an expression, write remote memory through assignment; single address space  
   - Example: OpenMP

   **Distributed Shared Memory:** Helps exploiting locality of references  
   - Simple statements  
   - Example: UPC

   b) Name the six basic functions in the message Passing Interface (MPI).

   - MPI_Init
   - MPI_Comm_rank
   - MPI_Comm_size
   - MPI_Send
   - MPI_Recv
   - MPI_Finalize

   c) Give the values of A[1] and B[0] in each process if the following MPI+C code was compiled and executed with 3 processes?

   ```c
   int i, j, my_rank, group_size, A[3], B[3], tag=1, root=0;
   MPI_Comm comm;
   MPI_Init (&argc, &argv);
   comm = MPI_COMM_WORLD;
   MPI_Comm_rank(comm, &my_rank);
   MPI_Comm_size(comm, &group_size);
   For (i=0; i<3; i++)
   {
      A[i] = B[i] = my_rank*group_size + i;
   }
   MPI_Scatter(A, 1, MPI_INT, B, 1, MPI_INT, root, comm);
   ```

   Here is the table of values for A[1] and B[0] in each process:

<table>
<thead>
<tr>
<th>Node 0</th>
<th>Node 1</th>
<th>Node 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[1]</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>B[0]</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
5. [a] Define temporal locality, spatial locality, sequential locality and give an example of what gives rise to each of them. What is the 90-10 rule?

Temporal locality – Future access will be into the same addresses that were accessed in the past. (Loops)

Sequential locality – if a[i] is referenced in the past, a[i+1] is likely to be referenced in the future. Instruction execution.

Spatial locality – if a[i] is referenced in the past, the a[i+δ] is likely to be referenced in the future. Arrays

90-10 rule – 90% of the references are into 10% of the space

b. Derive an expression for the average access time in an n-level memory hierarchy

the access freq. to Mi
\[ F_i = (1-h_1)(1-h_2)\ldots(1-h_{i-1})h_i \]

Effective access time
\[ T_{eff} = \sum F_i T_i \]
\[ = h_1 t_1 + (1-h_1) h_2 t_2 + (1-h_1)(1-h_2) h_3 t_3 + \ldots + (1-h_1)(1-h_2)\ldots(1-h_{n-1}) t_n \]

c. Does your expression work for both the write back and write through update policies? Explain.

It works in the case of reads in general. For writes, it works for write –back only. For write-through we operate at the speed of the slowest device.