MONASH University

Lecture Revision

OBJECTIVES

- Revise the materials and deepen the understanding of Week 4 to 5 lectures.

MARKS

- The applied session is worthed 3 of the unit final mark.

INSTRUCTIONS

1. **Preparation is required for this Applied session.** Please do NOT plan to complete the Applied preparation during the Applied class. **Late penalty** (5% per day) will apply if you submit late!
2. The purpose of these **flipped Applieds** is for students to get an opportunity to test their understanding of lecture/pre-reading material with a low cost in lost marks if they have not understood the material, rather than not understanding the material.
3. Students should produce **written answers** to the Applied questions prior to starting the Applied.
4. Students must submit their answers via Moodle in **PDF format** before their allocated class.
5. Students' answers will be reviewed in a question and answer format during the Applied. Each student will explain their answer.
6. For many of the questions, marks will be allocated for **both the correctness of the written answer and the correctness & clarity of the verbal answer.** Therefore, plan also for a brief 60 seconds summary explanation of each question. Try to focus on the most important and critical idea in the answer. Marks will be awarded on the student's ability to explain their submitted answers in the PDF: concise, focused and accurate answers will earn full marks, answers with errors and lengthy explanations will lose marks. Read the marking rubric for detailed instructions.
7. All questions are based on lecture/pre-reading slides, and lecture/pre-reading slides are in effect the answer sheets for these Applieds.
8. Worked answers will not be posted after Applieds as most of the answers are already in the lecture/pre-reading slides and discussed in the class.
9. Marks will not be awarded if you skip the class, or do not make any submissions, or make an empty submission to Moodle (unless a special consideration extension has been approved).
10. You are allowed to use search engines or AI tools to search for information and resources during pre-class preparation. However, search engines and AI tools are not allowed during the class.

MONASH University

Questions

Answer all the following questions.

Question 1 [SIMD & Vector Processing]:

Given a SIMD architecture machine with 4 processing elements. For the following piece of pseudocode performing array addition:

```
DO 10 I = 1,N

10 A(I) = B(I) + C(I)
```

A) If N = 7, draw a diagram (similar to W4 Lecture slides - page 16) to explain how the data is arranged in the memory for each processing element and how many loops are needed. Discuss if any processing elements will need to be disabled at any time.

For the following piece of pseudocode utilizing instruction for vector operations:

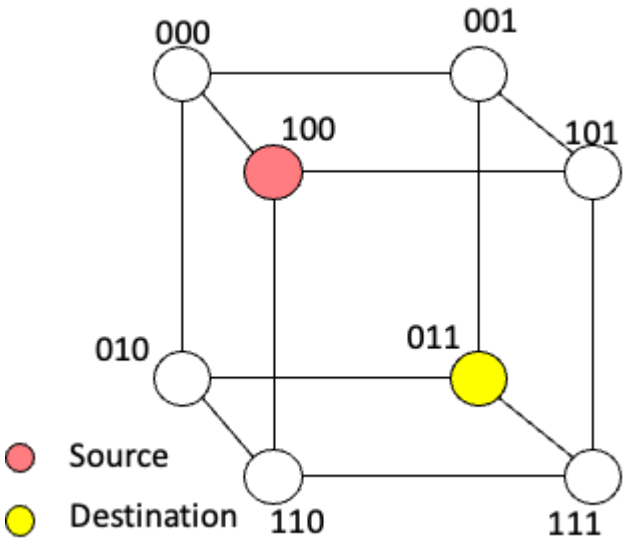
```
ADD N, A, B, C
```

B) Explain how the overheads of opcode fetches, decodes, and operand loads are reduced. Explain also how chaining is used in vector processing.

Question 2 [MIMD Architectures & Interconnection Topology]:

A) Compare and contrast Distributed Memory MIMD and Shared Memory MIMD. Provide at least two advantages & two disadvantages of Distributed Memory MIMD, and two advantages & two disadvantages of Shared Memory MIMD.

B) Given the following (hyper)cube interconnection topology, which organizes nodes with identifiers in binary numbers.



Explain how we can find the number of hops between nodes 100 and 011. Will the number of hops be decreased if we rearrange into a balanced binary tree topology?

Suppose one more node needs to be inserted, explain whether it is possible to keep using only 3 bits for identification. Will node insertion be easier for a binary tree topology?

MONASH University

Question 3 [Distributed Memory with Open MPI]:

Briefly discuss what is wrong with the following code snippet?

```
#include <mpi.h>
#include <stdio.h>

int main(int argc,char *argv[]) {
    int numtasks, rank, dest, source, rc, count, tag=1;
    char inmsg, outmsg='x';
    MPI_Status Stat;

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if(rank == 0) {
        dest = 1; source = 1;
        rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD,
    &Stat);
        rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
    }
    else if (rank == 1) {
        dest = 0; source = 0;
        rc = MPI_Recv(&inmsg, 1, MPI_CHAR, source, tag, MPI_COMM_WORLD,
    &Stat);
        rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
    }

    rc = MPI_Get_count(&Stat, MPI_CHAR, &count);
    printf("Task %d: Received %d char(s) from task %d with tag %d \n",
   rank, count, Stat.MPI_SOURCE, Stat.MPI_TAG);

    MPI_Finalize();

    return(0);
}
```