

# Advanced Computer Architectures

## ACA319

### Tutorial 1

week 2

---

### INTRODUCTION TO MPI

- Introduction to process model
- Introduction to message passing model
- Introduction to MPI library routines
- Historic background

Discuss "hello world" program:

```
#include <string.h>
#include "mpi.h"
int main(int argc, char **argv)
{ int myrank,nprocs;
  char text_out[5];
  char text_in[5];

  MPI_Status status;
  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
  MPI_Comm_size(MPI_COMM_WORLD,&nprocs);
  strcpy(text_out,"hello");
  if(myrank == 0) /*First process */
  { MPI_Send(text_out,5,MPI_CHAR,myrank+1,0,MPI_COMM_WORLD);
  }
  else if( myrank < (nprocs-1) ) /*All middle processes */
  { MPI_Recv(text_in,5,MPI_CHAR,myrank-1,0,MPI_COMM_WORLD,&status);
    MPI_Send(text_out,5,MPI_CHAR,myrank+1,0,MPI_COMM_WORLD);
  }
  else /*Last process */
  { MPI_Recv(text_in,5,MPI_CHAR,myrank-1,0,MPI_COMM_WORLD,&status);
  }
  MPI_Finalize();
}
```

# Advanced Computer Architectures

## ACA319

**Tutorial 2**

**week 3**

---

### USAGE OF MPI

- Introduction to collective communication
- Introduction to broadcasting
- Introduction to barrier synchronization
- Update on MPI control files

Discuss "partial sum" program:

```
#include <stdio.h>
#include "mpi.h"
int main(int argc, char **argv)
{ int myrank,nprocs;
  int partial_sum,sum,i;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD,&myrank);
  MPI_Comm_size(MPI_COMM_WORLD,&nprocs);
  for(i=myrank;i<=100;i+=nprocs)
  { partial_sum+=i;
  }
  MPI_Reduce(&partial_sum,&sum,1,MPI_INT,MPI_SUM,0,
  MPI_COMM_WORLD);
  MPI_Finalize();
}
```

# Advanced Computer Architectures

## ACA319

### Tutorial 3

week 4

---

### NETWORKS

#### QUESTION 1

- Construct a four-stage delta network with 16 inputs and 16 outputs by using elementary switches ('black boxes').
- How many stages are required for a delta network with  $n$  inputs and  $n$  outputs (assume  $n$  is a power of 2,  $n = 2^m$ ) ?
- How many elementary switches ('black boxes') are required for a delta network with  $n$  inputs and  $n$  outputs (assume  $n$  is a power of 2,  $n = 2^m$ )

#### QUESTION 2

- Prove that a shuffle-exchange network never requires more than  $2 * \log n - 1$  steps for a data exchange between two nodes.
- What is the similarity between a delta network and a shuffle-exchange network?
- What is the similarity between a hypercube and a PM2I network?

#### QUESTION 3

Give the (approximate) optimal values for  $m$  and  $a$  in a three level Clos network with  $N = 16,384$

#### QUESTION 4

The total number of interconnections in a three level Clos network is approximately

optimal when  $m$  is selected as follows:  $m \approx \sqrt{\frac{N}{2}}$

Prove this rule!

The University of Western Australia  
Electrical, Electronic and Computer Engineering  
A/Prof. Thomas Bräunl

# Advanced Computer Architectures

## ACA319

**Tutorial 4**

**week 5**

---

### **P-THREADS**

- Introduction to thread model
- Introduction to thread creation and termination
- Introduction to synchronization with mutex (semaphores)
- Historic background

# Advanced Computer Architectures

## ACA319

### Tutorial 5

week 6

### SEMAPHORES

1. With respect to the parallel program fragment below with two parallel processes:
  - a) Represent only the synchronization structure of this program as a Petri net!
  - b) Can deadlocks occur in the process system?

```
var critical: semaphore[1];  
k : integer;
```

*P1*

```
var a: integer;  
...  
loop  
  P(critical)  
  k:=a;  
  V(critical);  
end;
```

*P2*

```
var x,y: integer;  
...  
loop  
  P(critical)  
  k:=x;  
  V(critical);  
  y:=2*x-1;  
  P(critical)  
  k:=y;  
  V(critical);  
end;
```

2. Multiple identical processes need to access shared memory in a coarse-grained parallel MIMD system and should be synchronized with semaphores. A maximum of one process is allowed to access the shared data at any time.

- a) With which value should the semaphore be initialized?
- b) Complete the following program skeleton adding the missing synchronization operations:
- c) If the processes defined in part b) only read from the shared data, then these multiple processes could do so simultaneously. Change the program, so a maximum number of 5 simultaneously reading processes shall be allowed.

```
PROCESS work;  
VAR s: SEMAPHORE[...];  
BEGIN  
  LOOP  
    .....  
    (* Accomplish work on shared data *)  
    .....  
  END;  
END PROCESS work;
```

# Advanced Computer Architectures

## ACA319

### Tutorial 6

week 7

---

### MONITORS

Write a program in C or pseudo-code using pthreads for the parallel control of the data base for a warehouse. Since multiple processes can access the database simultaneously, their access must be synchronized through a **monitor**.

- Assume a data base with 1,000 items, specifying "item no.", "unit price" and "stock quantity"
- Assume a data base with 100 accounts, specifying "account no." and "budget"
- Use **condition variables** for
  - Waiting for a delivery if there is insufficient quantity in stock for a particular order
- The monitor requires entry functions for updating stock data in the form:
  - **New\_Stock\_Arrival** (item\_no, quantity)  
*Update the stock quantity of the particular item by adding the delivered amount.*
  - **Execute\_Order** (item\_no, quantity, account\_no)  
*Update the stock quantity of the particular item and deduct appropriate sum from specified account.*  
*Wait in condition variable if order quantity is higher than stock quantity for desired item.*

**Note:** There are two possible solutions for this. You can either use a single condition variable for all items, or use one condition variable *per item*. Specify which **signal** version ("free 1" or "free all") has to be used with either implementation.

# Advanced Computer Architectures

## ACA319

**Tutorial 7**

**week 8**

---

### **SORTING**

Introduction to parallel sorting techniques. Look up the definitions of Odd-Even Transposition Sort (OETS) and Parallel Bucket Sort (PBS).

#### **QUESTION 1**

Sort the following numbers using OETS:

2, 1, 4, 6, 4, 9, 7, 8, 3, 1, 6, 5, 3, 5, 3, 1

#### **QUESTION 2**

Sort the following numbers using PBS:

2, 1, 4, 6, 4, 9, 7, 8, 3, 1, 6, 5, 3, 5, 3, 1

#### **QUESTION 3**

Discuss the difference in time complexity between OETS and PBS in terms of number of elements to be sorted (N).

# Advanced Computer Architectures

## ACA319

**Tutorial 8**

**week 9**

---

### **SIMD and SPMD**

Introduction to SIMD (and SPMD) programming style.

- SIMD programming in Parallax-3
- SPMD programming in MPI

### **FRACTAL**

Write a sequential program in C or pseudo-notation to generate a 1D fractal curve using mid-point displacement.



Translate this sequential program into a SIMD or SPMD program using C with MPI.

# Advanced Computer Architectures

## ACA319

**Tutorial 9**

**week 10**

---

### PARALLEL PERFORMANCE

#### QUESTION 1

A parallel program is to be executed on a MIMD computer with 1,000 processors.  
50% of all commands can be parallelized on all PEs  
50% of all commands cannot be parallelized and have to run sequentially.  
What is the speedup of this program for this computer system?

#### QUESTION 2

A parallel program is to be executed on a SIMD computer with 1,000 processors.  
50% of all commands can be run vectorally on all PEs  
50% of all commands are scalar (sequential) commands  
What is the speedup of this program for this computer system?

#### QUESTION 3

A parallel program is to be executed on an MIMD computer with 100 processors. However, the following restrictions apply:

- 2 % of all commands during program execution are executed sequentially,
- 20 % of all commands can only be executed on 50 processors,
- the rest can be executed on all available processors.

What is the speedup of this program for this computer system?

#### QUESTION 4

A parallel program is to be executed on a SIMD computer with 10,000 processors.  
Measurements show that on the average the PEs were active for 30% of the run time and were inactive the rest of the time (e.g. due to parallel IF–THEN–ELSE statements).  
What is the speedup of this program for this computer system?

# Advanced Computer Architectures

## ACA319

**Tutorial 10**

**week 11**

---

### **DATA DEPENDENCIES and AUTOMATIC VECTORIZATION**

#### **QUESTION 1**

Determine all of the data dependences of the statement sequence:

```
S1:  A := B + C;  
S2:  B := A + C;  
S3:  D := B * C - 2;  
S4:  A := B / C;
```

#### **QUESTION 2**

Determine all (re-) ordering requirements that follow from question 1.

#### **QUESTION 3**

Determine all of the data dependences with directions.

```
for (i=1; i<n-1; i++)  
{ S1:  A[i] := B[i] + C[i];  
  S2:  B[i-1] := 2 * D[i] + 1;  
  S3:  C[i] := A[i] + B[i];  
  S4:  E[i] := A[i+1] / 7;  
}
```

#### **QUESTION 4**

Determine all (re-) ordering requirements that follow from question 3.

# Advanced Computer Architectures

## ACA319

**Tutorial 11**

**week 12**

---

### **AUTOMATIC PARALLELIZATION**

#### **QUESTION 1**

Determine all of the data dependences with directions.

```
for (i=4; i<n; i++)  
{ S1: A[i-1] := 2 * B[i] + 3;  
  S2: B[i]   := 2 * D[i] + 1;  
  S3: E[i]   := E[i] + 5;  
  S4: C[i]   := A[i] + D[i];  
}
```

#### **QUESTION 2**

Determine all of the dependences that must be synchronized.

#### **QUESTION 3**

Parallelize the loop (in pseudo-notation '**doacross**') for a MIMD system. Attempt to achieve maximum parallelism!

The University of Western Australia  
Electrical, Electronic and Computer Engineering  
A/Prof. Thomas Bräunl

# Advanced Computer Architectures

## ACA319

**Tutorial 12**

**week 13**

---

### **EXAM PREPARATION**

General repeat and question & answer session.