# EECC 756 - Spring 2002
# Homework Assignment #1, Due April 16

1. The following code segment, consisting of six instructions, is to be executed 64 times for the evaluation of vector arithmetic expression: D(I) = A(I) + B(I) X C(I) for $0 \leq I \leq 63$

```
Load R1,B(I)         /R1  ← Memory(α + I)/
Load R2, C(I)        /R2  ← Memory(β + I)/
Multiply R1, R2      /R1  ← (R1) X (R2)/
Load R3,A(I)         /R3  ← Memory(γ + I)/
Add R3, R1           /R3  ← (R3) + (R1)/
Store D(I), R3       /Memory(θ + I) ← (R3)/
```

Where R1, R2, R3 are CPU registers, (R1) is the content of or R1, and α, β, γ, θ, are the starting memory addresses of arrays B(I), C(I), A(I) and D(I) respectively. Assume four cycles for each Load or Store, two cycles for the Add, and eight cycles for the Multiply on either a uniprocessor or a single processing element (PE) in an SIMD machine.

a) Calculate the total number of CPU cycles needed to execute the above code segment repeatedly 64 times on an SISD uniprocessor computer sequentially, ignoring all other delays.

b) Consider the use of an SIMD computer with 64 PEs to execute the above vector operations in six synchronized vector instructions over 64-component vector data. Calculate the total execution time in cycles on the SIMD machine, ignoring instruction broadcast and other delays.

c) If both the SISD and SIMD machines run at the same clock speed, what is the speedup gain of using the SIMD computer over the SISD computer?

2. Compare the PRAM models with physical models of real parallel computers in each of the following categories:

a) Which PRAM variant can best model SIMD machines and how?

b) Repeat the question in part (a) for shared-memory MIMD machine.

c) Criticize the inadequacy of the PRAM to model most real parallel computers.

3. Develop an algorithm for a CRCW PRAM with *n* processors for multiplication of two *n* x *n* matrices. Find the time complexity of your algorithm.

4. Analyze the data dependencies among the following statements in a given program:

```
S1:    Load R1, 1024         /R1 ← 1024/
S2:    Load R2, M(10)        /R2 ← Memory(10)/
S3:    Add R1, R2            /R1 ← (R1) + (R2)/
S4:    Store M(1024), R1     /Memory(1024) ← (R1)/
S5:    Store M(R2), 1024     /Memory(64) ← 1024/
```

Where (Ri) means the contents of register Ri and Memory(10) has 64 initially

a) Draw the dependence graph to show all dependencies.
b) Are there any resource dependencies if only one copy of each functional unit is available to the CPU?
c) Repeat the above for the following program statements:

```
S1:    Load R1, M(100)       /R1 ← Memory(100)/
S2:    Move R2, R1           /R2 ← (R1)/
S3:    Inc R1                /R1 ← (R1) +1/
S4:    Add R2, R1            /R1 ← (R2) + (R1)/
S5:    Store M(100), R1      /Memory(100) ← (R1)/
```

5. A sequential program consists of the following five statements, S1 through S5. Considering each statement as a separate process, clearly identify input set $I_i$, and output set $O_i$ of each process. Restructure the program using Bernstein's conditions in order to achieve maximum parallelism between processes. If any pair of processes cannot be executed in parallel, specify which of the three conditions is not satisfied.

```
S1:    A = B + C
S2:    C = B X D
S3:    S = 0
S4:    Do I = A, 100
           S = S + X(I)
       End Do
S5:    IF (S > 1000) C = C x 2
```

6. Consider the execution of the following code segment consisting of seven statements. Use Bernstein's conditions to detect the maximum parallelism embedded in this code. Justify the portions that cannot be executed in parallel and the remaining portions that must be executed sequentially. Rewrite the code using parallel constructs such as ***Cobegin*** and ***Coend***. No variable substitution is allowed. All statements can be executed in parallel if they are declared in the same block of a (***Cobegin***, ***Coend***) pair.

```
S1:    A = B + C
S2:    C = D + E
S3:    F = G + E
S4:    C = A + F
S5:    M = G +C
S6:    A = L + C
S7:    A = E + A
```

7. Let $\alpha$ be the percentage of a program code which can be executed simultaneously by $n$ processors in a parallel computer system. Assume the remaining code must be executed sequentially by a single processor. Each processor has an execution rate of $x$ MIPS, and all processors are assumed equally capable.

a) Derive an expression for the effective MIPS rate when using the system for the exclusive execution of this program, in terms of $n$, $\alpha$, and $x$.
b) If $n = 16$ and $x = 4$ MIPS, determine the value of $\alpha$ which will yield a system performance of 40 MIPS.

8. The following Fortran program is to be executed on a uniprocessor and a parallel version is to be executed on a shared-memory MIMD multiprocessor:

```
L1:              Do 10  I = 1, 1024
L2:                  SUM(I) = 0
L3:                   Do 20 J = 1, I
L4      20                SUM(I) = SUM(I) + J
L5:     10       Continue
```

Suppose statements 2 and 4 each take two machine cycles, including all CPU and memory-access activities. Ignore the overhead caused by the software loop control (statements L1, L3, and L5) and all other system overhead and resource conflicts.

a) What is the total execution time of the program on a uniprocessor.
b) Divide the I-loop iterations among 32 processors with pre-scheduling as follows: Processor 1 executes the first 32 iterations (I = 1 to 32), processor 2 executes the next 32 iterations (I = 33 to 64), and so on. What is the execution time and speedup factor compared with part (a) (Note that the computational workload dictated by the J-loop is unbalanced among the processors.
c) Modify the given program to facilitate a balanced parallel execution of the computational load over all 32 processors.
d) What is the minimum execution time of the modified balanced program of part (c) when executing on 32 processors? What is the new speedup over the uniprocessor?

9. Exercises: 2.4, 2.5, 2.6, 3.2, 3.12, 3.13, 3.17 in the textbook “Parallel Computer Architecture”.