Problem set 1, Part 1

1 Theory

1. List and describe the six computer architectures classified in Flynn's taxonomy

- Single Instruction Stream, Single Data Stream (SISD)
  - Computer with no parallelism. Based on sequential operations with executions on single instruction streams. SISDs uses a single uni-core processor
- Multiple Instruction Stream, Single Data Stream (MISD)

  MISD is a computer architecture exploring parallelism. In MISD multiple instructions operate on one single data stream. MISD is not widely used, but its fault tolerant abilities have lead to uses in system like the Space Shuttle flight control computer.
- Single Instruction Stream, Multiple Data Stream (SIMD)
  - SIMDs is an architecture which can operate on multiple data stream simultaneously with one single operation. The same operation is performed on the different data stream in parallel. First uses of SIMDs was in vector supercomputers in the 1970s. Most modern CPUs can use SIMD architecture.
- Multiple Instruction Stream, Multiple Data Stream (MIMD)

  MIMD is an architecture which fully exploits parallelism. In MIMD multiple processors can functions asynchronously and independently. This architecture allows executing different instructions on different data. MIMD computers may be either shared memory or distributed memory machines.
- Single Program, Multiple Data Stream (SPMD)
  - SPMD is a subcategory of MIMD. In SPMD different tasks are split up on multiple processors with different input simultaneously achieving fast and parallel processing of data. SPMD is the most common style of parallel programming.
- Multiple Programs, Multiple Data Stream (MPMD)
  - In MPMD processors are simultaneously running at least 2 independent programs.

2. You want to parallelize the work done in the loop below. It turns out that the work done by do stuff() is inherently serial for iterations i = 0–250, but you are able to fully parallelize the work done on iterations i = 251–999. Describe the possible speedup in terms of Tparallel, Tserial and p, according to Amdahls law.

```
for(int i = 0, i < 1000, i++) {
  do_stuff()
}
```

Tserial = 250

Tparallel = 750

p = Tparallel/(Tserial + Tparallel) = 0.75

Parallelizing 75 % of the for - loop will possibly give a system speed up dependent on the speed up factor of parallelizing as shown in Amdahls law

S_latency(s) = 1 / ((1 - p) + (p / s)) (1)

p : Fraction of system enhanced

s: Enhancement factor

Computing S_latency for ranging values of s will show that the system speed up will converge as the speed up grows as the second part of the equation (p/s) under the fraction in (1) will diminish towards zero and the first part ((1-p)) will dominate the system speed up.

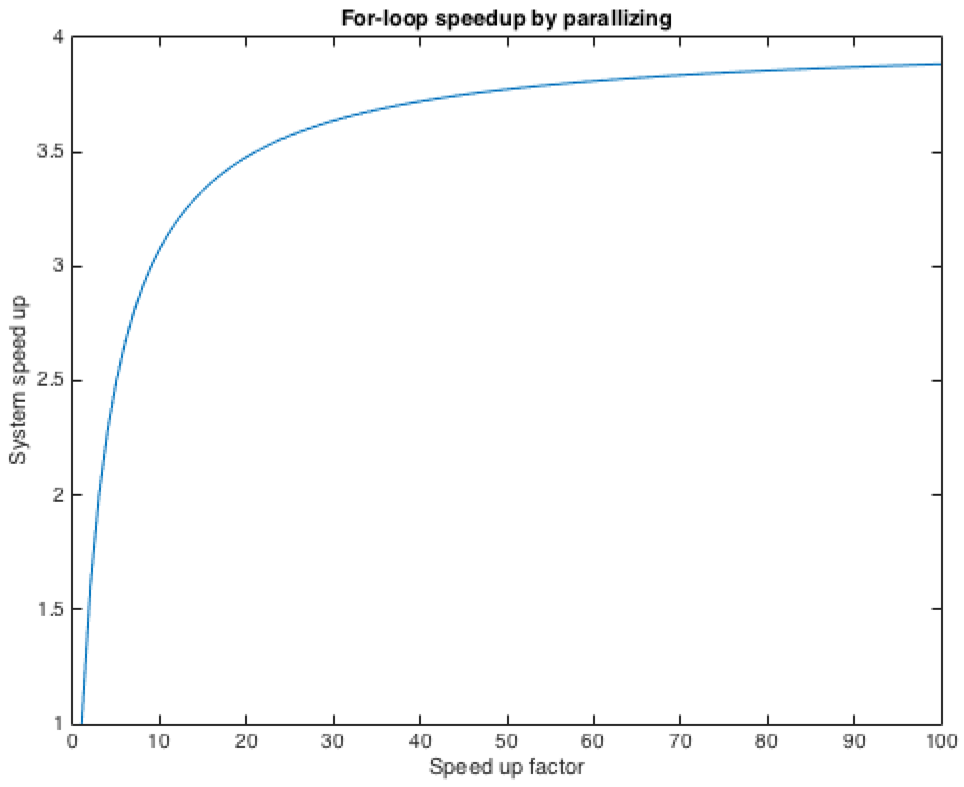
(Matlab simulation shown on next page)

```
% Selection of speed ups from 1 to 100 times
s = [1:1:100];

% 75 percent of for loop enhanced
p = .75;

% The speed up evaluated using amdahls law with different values for
 speed
% up
sLatency = 1 ./ ( ( 1-p ) + ( p ./ s ) );

plot(sLatency),title('For-loop speedup by parallizing'),xlabel('Speed
 up factor'),ylabel('System speed up')
```



Published with MATLAB® R2015a

3. What is the difference between parallel computing and distributed computing (according to Pacheco)

Parallel computing: Parallel computing allows a program to contain multiple tasks which cooperate closely to solve a problem.

Distributed computing: Distributed computing allows a program to cooperate with other programs to solve a problem.

A clear-cut distinction between parallel and distributed programs do not exist, but a parallel program usually runs multiple tasks simultaneously on cores that are physically close to each other and that either share the same memory or are connected by a very high-speed network.

Distributed programs, on the other hand, tend to be more “loosely coupled”.

(From *An Introduction to Parallel Programming*)