EEE4084F QUIZ 3 2014 - ANSWERS

Q1 (b)

Q2 (a) Common design patterns (any two of these accepted):
- Pipeline
- Replicate & Reduce
- Repository
- Divide & conquer
- Master/slave
- Work queues
- Producer/consumer flows

(b) Please see descriptions and diagrams in Lecture 11

Q3 *Barrier synchronization:*

<u>All</u> tasks usually involved; Each task does work until it reaches the barrier, and then blocks.

When the last task reaches the barrier all the tasks are synchronized. After this the tasks are automatically released to continue their work… the programmer usually decides at what point barrier sync happens.

Example program:

Searching for a the number of times a string is found can be done by splitting the data to search into N blocks done by N threads. All threads do searching in their blocks, a barrier is placed that forces all threads to complete before the number of occrences of the string is displayed by the first thread.

Locking synchronization:

May concern <u>any number</u> of tasks, <u>not</u> necessarily all (depends on where in the code tasks are, some may continue doing stuff while others wait turn for access).

Usually used to serialize/protect access to global data or critical section of code. Only one task at a time may have the lock / semaphore. Tasks can attempt to get the lock need to wait for the task that has the lock to release it, granted on a FCFS basis.

Usually synchronization is blocking (ideal untill ready), but could be non-blocking (i.e., do other work until lock is available).

Example program:

A lookup table T[x] for sin(x) is stored in an EEPROM on a RAM-limited embedded system, and will be used later to quickly interpolate sin values without calculating them numerically. Generation of T is split between N threads, each write to the EEPROM to save a T[x] value is done in a critical section locked by a semaphore.

Q4

Q5(a) A data dependency is caused by different tasks accessing the same variables (i.e., memory addresses).

Q5(b) Loop carried data dependence

dependence between statements in different iterations

Loop independent data dependence

dependence between statements in the same iteration

Lexically forward dependence:

source precedes the target lexically

Lexically backward dependence:

opposite from above

Right-hand side of an assignment precede the left-hand side

Q5(c) Common approaches to work around data dependences tend to depend on the type of data used in the system...

For distributed memory architectures:

- use of synchronization points (periods when sets of shared data is communicated between tasks).
- for shared memory architectures: make use of read/write synchronize operations (no sending of data, just temporarily block other tasks from reading/writing a variable).

Q6 (a) The offset error of an ADC (similar to the offset error of an amplifier) is defined as a deviation of the ADC output code transition points that is present across all output codes.

(b) ENOB = (SINAD - 1.76)/6.02 = (68 - 1.76)/6.02 = 11 bits

(c) Easy, a stadard flash ADC has 2^N-1 comparators, so 255 would be needed.

Q7 iv -- ha ha :) tried to make it difficult with suggestions like the elucidation quotient (EQ) being a bit like IQ.