EECS 473, Homework #1

Due 10/11 at 11pm. This should be good preparation for the midterm, you should also look at old exams on the website when studying for the exam. In particular, expect a design question much like those found in the prior exams. I expect this will take about 6 hours to do.

1. Everything, Everywhere, All at Once

Q1. Multiple choice—Circle the best answer. **[12]**

a) Which of the following is an advantage of RM scheduling over EDF scheduling?

- ***RMS handles deferred interrupts properly (as long as priority inheritance is enabled) while EDF often does not.RMS can schedule certain sets of periodic tasks EDF cannot.***
- ***RMS doesn't require dynamic priorities for periodic tasks, while EDF generally does.***
- ***When RMS fails to schedule, it will always fail to schedule the task with the largest CPU needs, while EDF could fail to schedule other tasks.***

b) Which of the following statements about linear regulators is FALSE?

- ***A 5V output linear regulator with a 10V input would be expected to waste less power than that same regulator with a 12V input.***
- ***An LDO is a type of linear regulator.***
- ***A linear regulator is generally expected to have a less-noisy output than a switching regulator.***
- ***A real linear regulator with a 10V input and 5V output would be expected to waste no more than 50% of the input power.***

c) Which of the following statements about the GPL is FALSE?

- ***It stands for the GNU Public License.***
- ***Things which use the GPL are open source and in the public domain.***
- ***The GPL is sometimes referred to as being a "viral license" because if you use code licensed under the GPL you likely need to license that code under the GPL.***
- ***Linux and gcc are both licensed under the GPL.***

d) The term "rat's nest" is best understood to refer to what with respect to PCBs?

- ***The layout when most/all connections are still "air wires".***
- ***The schematic once the connections have been made.***
- ***The (potentially excessive) use of "neck downs" to reduce PCB trace resistance.***
- ***The use of an excessive number of vias on a PCB.***

Q2. Say you have the following groups of tasks. For each group find the CPU utilization. Figure out which groups are RM schedulable. ***Only*** where needed are you to show the critical instant analysis. **[12]**

Group	*T1 Execution Time*	*T1 Period*	*T2 Execution Time*	*T2 Period*	*T3 Execution Time*	*T3 Period*	*CPU Utilization?*	*RM Schedulable?*
A	2	7	2	7	2	7	6/7	Yes (see below)
B	1	10	5	12	4	21	0.707	Yes
C	4	8	3	9	5	27	1.019	No
D	1	3	2	5	2	8	0.983	See below

For three tasks, if the total CPU utilization is less than .779 it's schedulable.

Group A

Need critical instant analysis. Given T1-T3 all have the same priority, they could go in a lot of orders, but all will finish by the start of time 6 so all meet their deadline.

Task	0	1	2	3	4	5	6
T1							
T2							
T3							

The tasks schedule under RMS.

Group D

Need critical Instant analysis. Task3 doesn't meet its deadline.

Task	0	1	2	3	4	5	6	7	8	9
T1										
T2										
T3										

Q3. Consider an embedded application which consists of 3 tasks named A, B, and C. Each task is CPU bound (that is, there is no I/O or memory operations which take significant time to execute) and periodic. Each task must complete before the next instance of that task is ready to start. These tasks have the following properties and requirements. You are to assume there is no overhead of any type (including scheduling overhead) and that this machine runs any given instruction in exactly the same amount of time.

Task	**Max. number of instructions executed**	**Period**
A	1 million	100ms
B	2 million	150ms
C	4 million	500ms

a) Given a choice of a processors with MIPS ratings of 25, 30, 35, and 50, which is the slowest that will be able to schedule these tasks using EDF? Show your work. **[5]**

For a. we just need the CPU utilization to be no more than 100%.

- A needs to execute 10 MIPS.
- B needs to execute 2/.15= 13.3 MIPS
- C needs to execute 8 MIPs.

So we need at least 31.7 MIPS. So 35 MIPS will work.

b) As part a but for RMS. **[10]**

- 31.7/50=63.4%, so 50 MIPS will work for sure.
- 31.7/35 ~=90%, so maybe.
- 30 and 25 MIPS are too small.
- For 35, task A will run for under 29ms. Task B for under 58ms and task C for under 115ms. That means in the first 500 clocks, task A will run for 29ms*5, task B for under 58ms*4. That leaves 123ms for task 3 to run. That's enough time, so yes it will work!

Q4. Answer the following: **[10]**

a. Provide two tasks that together are not schedulable under Round Robin where one has exactly a 10% CPU utilization and the total CPU utilization is less than 75% or state why no such tasks exist.

Task	Runtime	Period
A	60 seconds	100 seconds
B	100 seconds	1,000 seconds

When first launched, task A will get 50% of the CPU and task B will get the other 50%. By time 100 task A will only have run for 50 seconds and it's deadline will be missed.

b. Provide two tasks that together are schedulable under Round Robin where one has exactly a 10% CPU utilization and the total CPU utilization is greater than 75% or state why no such tasks exist.

Task	Runtime	Period
A	1 seconds	10 seconds
B	700 seconds	1,000 seconds

Task A will trivially finish. In the 1000 seconds after release, task A will take up only 100 seconds of runtime so B will have plenty of time to run.

Q5. Let's look back and see how EDF does with the scheduling problems that RM struggles with.

a. Consider the tasks found in group D of problem 1.

i. Show how EDF would schedule the tasks through time 15 if all three were released at time 0. **[4]**

ii. Explain how having dynamic priorities is helpful here. **[3]**

Below is how EDF would schedule things. The number in the box is when the job is due

Task	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T1	3			6			9			12			15			18
T2		5	5					10	10		15	15				
T3					8	8								16	16	

In some cases T3 needs to have a higher priority than T2 (e.g. time 5) and other times it is important that T2 have a higher priority than T3 (e.g. time 1)

b. Consider the tasks found in group C of problem 1. Either show at what time a task would first fail to make its deadline OR show that EDF could schedule up to time 40 (whichever is true) assuming all tasks were first released at time zero. If it schedules through time 40, will it ever fail at scheduling? Justify your answer. **[8]**

OK, this is way tedious. We'll give you points for pretty much anything, just too much work. The point is it *will* fail, it's just not fun proving it. But since it needs more than 100% of the CPU, it must fail at some point.

Q6. Consider the following set of jobs:

i. Task A has a period of 10ms, a worst-case execution time of 3ms of CPU time, and needs to use resource "X" for the entire time.

ii. Task B has a period of 21ms and a worst-case CPU time of 5ms

iii. Task C has a period of 33ms, a worst-case CPU time of 5ms, and needs to use resource "X" for the entire time.

a. If there were no resource dependencies, would the above tasks be schedulable under RMS? EDF? Justify your answer. **[4]**

- yes, 3/10+5/21+5/33 is about 69%. RMS and EDF can schedule that since RMS can schedule anything three tasks that have a utilization less than 0.779.

b. With the dependencies, but assuming no priority inheritance, show an example of RMS and EDF both failing. Assume that task "A" will not even start if task C is running (as resource X isn't available) and via versa. **[8]**

- Assume task C released 0ms and tasks A and B are released at 1ms. In that case, task A doesn't finish on time (note, task A doesn't start until time 9 because C is running).. By task A's deadline (11ms) it has only run for 1ms.

Task	0-1	1-2	2-3	3-4	4-5	5-6	6-7	7-8	8-9	9-10	10-11
TA											
TB											
TC											

c. Explain how priority inheritance addresses the problem of priority inversion in the general case. **[5]**
   - In the general case it prevents a high priority task from being blocked by a lower priority task (as happened here with B running before A).

d. Will the above tasks be schedulable if priority inheritance is in use? Consider both EDF and RMS. **[6]**
   - This is pretty hard to prove. The basic argument here is to consider each task.
     - The worst case for A is that C blocks A and that can't last more than C's run time (5) and that will leave plenty of time for A to resolve (3) before its deadline (10).
     - The worst case for B is that C's promotion above B could cause problems. Worst would be that when C runs it is always promoted above B. Even then the worst case is that in B's 20ms period, C runs once and A runs twice. That's 5ms*2+2ms or 12ms. That's plenty of time for B to finish (5ms run time, means it will certainly finish by 17ms after it is released).
     - The worst case for C is the same as the normal critical instance as it has the lowest priority. And we already know C schedules in that case.

   We'll take a lot of answers here as long as you formed a reasonable argument.

Q7. In your own words answer the following questions:

a. What does it mean when a license is said to be "viral"? **[4]**
b. Which of the Creative Commons licenses are viral? **[2]**
c. What is the difference between the GPL and the LGPL? **[2]**
d. Describe the CC BY-NC-SA license. **[3]**

a. A viral license is a generally negative term for a license that not only applies to the current licensee, but also to some or all of the work of that licensee. In the free software arena, where the term "viral license" is most commonly used, it generally means that the licensee is agreeing to attach the license to any derivative work under the same license. In that way the license can be aid to have "infected" all derivative work. The term is a pejorative and free software supporters sometimes object to the word. Other supporters, like your instructor, think it's a very apt description of the license. ☺
b. Anything with "SA" (Share alike) would count as viral. You can make an argument that they are all viral. So, for example, CC-BY would require everyone who uses your work, directly or indirectly, to attribute it to you. However, they aren't specifically required to use the CC-BY. We'll take that as an answer because "viral" in this context isn't hugely well defined. But that's not the typical meaning.
c. The LGPL allows a work (program) to be used by a program without requiring that the program using the LGPLed library be licensed under the GPL or LGPL.
d. You can use the material for non-commercial reasons and only as long as you provide attribution to the creator(s). If you distribute those changes, you must license it under the CC BY-NC-SA license.

Q8. Read the Wikipedia article on Tivoization. Explain in your own words:

a. What Tivo is/was and its basic business model. **[2]**
b. What Tivo did with respect to the GPL. **[2]**
c. How the GPLv3 is a reaction to it. **[4]**
d. What are the major changes from GPLv2 to GPLv3 other than what you mentioned in part b? **[4]**

a. TIVO was a digital video recorder. People paid for the DRV itself and then paid a service fee which enabled them to easily tape TV shows.
b. Tivo created a software for their DVRs using software incorporated with copyleft license GPLv2, due to which they were mandated to provide the source code of the software whoever asked them. To maintain their hold on the product while complying with the GPLv2, they added a mechanism in the product (digital signature) which would ensure that only their own software can work with their hardware and not the modified code.
c. GPLv3 was created with keeping the above loophole in mind. GPLv3 requires "user products" include "installation information" including keys needed to be able to run (potentially modified) GPL code.
d. We'll take mentions of any two of these three ideas:
    - Changes to make it easier to combine GPL code with other code.
    - Addition of a patent licensing agreement allowing people to use code even if the author has relevant patents.
    - The notion that code run on a remote server need not be provided to everyone that uses the code.

Q9. Answer the following questions. You may need to use resources beyond the lecture material.

a. Jitter **[8]**
    i. What is jitter in the context of an RTOS?
    ii. Say you have an aperiodic task that always takes 1ms of CPU time to run and is triggered on an interrupt (say a GPIO pin going high). Now say you have a task that disables interrupts for no more than 10ms.
        1. What is the worst-case response time the system will have to the interrupt?
        2. Assuming this is the only source of jitter, how much jitter might we have?
b. In the C language, what is a "void *" and why does FreeRTOS use it? Provide an example. **[4]**
c. Sometimes a variable is declared as "volatile". **[4]**
    i. Explain why we often have MMIO locations declared as volatile. What might happen if we didn't do so.
    ii. You hear someone say: "The volatile keyword is often thought to have something to do with caches—that isn't the case. It is directed to the complier, not the hardware." Explain what they mean.
d. Define the term "critical section". **[2]**

i. The term “top half” and “bottom half” (sometimes “upper-half” and “lower-half”) is often used when discussing deferred interrupts.
   1. Explain that terminology.
   2. Which “half” of the interrupt is most likely to cause additional jitter for other interrupts? Explain your answer.

ii. When using deferred interrupts we use a semaphore, but not to lock a resource as we might for a shared device. Explain how and why we use a semaphore here. (Your answer will likely be several sentences.)

a.

I. Jitter is variability in response (or release) time for a given operation. It’s generally associated with delays in interrupt response time.

II. .
- The interrupt may take 11ms to run (or 10ms to start).
- The variation can be from 0ms to 10ms, so 10ms.

III. There are many potential answers here. We’ll take anything reasonable
- For interrupts, the hardware may be caught in different states at different instances of the interrupt. So, for example, if the CPU is in the middle of something like a “move multiple registers” (push and pop on ARM for example) command that takes a while to run, can’t be aborted and needs to be atomic, the interrupt won’t be able to process until the instruction finishes.
- For a scheduled periodic task, jitter can exist due to aperiodic tasks (often interrupts) interfering.

b. “void *” is a way to generically refer to a pointer and it can be converted to any other pointer type without an explicit cast (in C, C++ requires the cast). So why use it (especially in C++)? It tells the reader that some pointer shenanigans will be happening and to watch closely for them.

FreeRTOS uses them in a number of places. The one we’ve probably seen most is in tasks where you can pass pointer to an arbitrary data structure. When you call xTaskCreate, one of the arguments is a pointer to an arbitrary data structure that the task can then use. The void * is highlighted.

```
BaseType_t xTaskCreate( TaskFunction_t pvTaskCode,
                        const char * const pcName,
                        unsigned short usStackDepth,
                        void *pvParameters,
                        UBaseType_t uxPriority,
                        TaskHandle_t *pxCreatedTask );
```

c. .

I. When reading or writing MMIO locations we want the address to be read or written, not optimized away (either to a register or just removed entirely) by an optimizing complier. The volatile keyword makes it so the complier will not remove or otherwise optimize a way any reference to that location.
Say we didn’t declare an MMIO location, XX, to be volatile. In that case, a command like “while(XX==0);” would likely just result in a single read to XX (because the complier “knows” that it isn’t changing the value and nothing else

could be). If XX is waiting on a status register to change, that would be a problem.

II. There seems to be a common belief that the volatile keyword tells the processor to not use the cache for the data access—that it requires things to go "all the way to memory". That isn't the case, it just tells the complier not to optimize away accesses to that memory location.
    - Note: The complier doesn't generally (ever?) control things like that. The choice of what can be cached or not is often fixed by address (in most low-end microprocessors) or requires changing some tables associated with memory in the hardware (on many high-end processors).

d. A section of code where it could be problematic if another (often specific) task was running at the same time as this code. So, we lock out (some) other code from running at that time. Often used to make sure only one task is accessing a shared resource at any given time. (There are a lot of similar definitions of course).

Q10. Deferred interrupts and Mutexes

a. Explain the advantages of using a "deferred interrupt". **[4]**
b. The term "top half" and "bottom half" (sometimes "upper-half" and "lower-half") is often used when discussing deferred interrupts. Explain that terminology. **[4]**

a. It allows us to have code that is in response to an event without having that code necessarily have the highest priority.
b. .
    - The "top half" is the ISR code. The "bottom half" is the task that the ISR causes to be run.
    - The top half. While code is running their other ISRs probably can't be run (some might be able to if you allow interrupt preemption)

We initialize the semaphore as not being available. When the "bottom half" first runs, it waits on the semaphore. When the "top half" wants to cause the bottom half to run it releases the semaphore. The bottom half can now be scheduled to run. Once it's done its job, the bottom half waits in the semaphore again.

Q11. Say we have the following datatypes and functions in a pseudocode-like language:

```
semaphore SemaphoreDeclare(state init);
void SemaphoreTake(semaphore);
void SemaphoreGive(semaphore);
```

where state is a two-state type with states FREE and BUSY

a. Consider the following code in this pseudocode-like language:

```
Init:
  semaphore a=SemaphoreDeclare(BUSY);
  int i=0; j=0;
  int x[8]=0; //all values in the array initialized to 0.

ISR:
  i++;
  SemaphoreGive(a);

Task1:
  SemaphoreTake(a);
  if(i>7) // Change made on 10/9.
      {i=0; j++}
      x[i]=j;
```

i. Why was the semaphore initialized to BUSY? What would happen if it were instead initialized to FREE? **[3]**

ii. What would be the value of array x, i , and j after the ISR was called 20 times? **[7]**

- It was initialized to BUSY because we didn't want Task1 to do its thing until after the interrupt had occurred. If it were initialized to FREE, Task1 could have done its thing and it would be as if one extra interrupt had happened. In this case that's not a big deal, but if the ISR were, say, supplying data to the task, then the task would run when there was no data. That would likely result in an error of some sort.
- x would have the value {2, 2, 2, 2, 2, 1, 1, 1}, j=2, i=4 ←

Note the above is a correction from when first posted.

b. Say you have a function named "work()" that takes some time and in which only one task is allowed to run at a time. Using a similar format to what we have in part a, write an Init, Task1, and Task2 function which uses a Mutex to insure that Task1 and Task2 are never both running "work()" at the same time. (Use the function names as above but replace the work "Semaphore" with "Mutex" in all cases). **[10]**

```
Init:
      mtx = semaphoreCreateMutex()
Task1:
   MutexTake(mtx);
   work();
   MutexGive(mtx);
Task2:
   MutexTake(mtx);
   work();
   MutexGive(mtx);
```

Q12. Consider the following code found as the read function member of the file_operations struct for a Linux kernel module. It is associated with the device file "/dev/txx2" (so a read of the file /dev/txx2 will result in this function being called). Assume that everything is set up appropriately beforehand. Ignore the fact that copy_to_user's return value is being ignored (it's just a warning…).

```
const char s[] = ,"TheABCsofDRL";
ssize_t memory_read(struct file *filp, char *buf,
                    size_t count, loff_t *f_pos) {


    if(*f_pos>=6)
         return 0;
    int x=count>2?2:count;
    /* Transferring data to user space */
    copy_to_user (buf, s+(*f_pos+1), x);
    printk("<1> *fpos= %d\n",*f_pos);
    /* Changing reading position as best suits */

    *f_pos+=x;
     return x;
}
```

a. Say a user program is run that opens the file and then reads 1 byte at a time until it finds the end-of-file. Each time it reads the data, it immediately prints what it reads.
   i. What will appear in the log file? **[5]**

<1> *fpos=0
<1> *fpos=1
<1> *fpos=2
<1> *fpos=3
<1> *fpos=4
<1> *fpos=5

   ii. What will be printed? **[5]**

heABCs

b. Say that someone does a cat of /dev/txx2.
   i. What will appear in the log file? **[5]**

<1> *fpos=0
<1> *fpos=2
<1> *fpos=4

   ii. What will be printed? **[5]**

heABCs

Q13. In your own words, define the following terms in the context of PCB design: **[8]**

a. Power Integrity (PI)

Power integrity is the goal of maintaining a desired voltage from "power" to ground at various parts of the board/PCB.

b. Signal Integrity (SI)

Signal Integrity is the analysis of the system to ensure that the signal carried on the wires/traces do not lose their value or gets corrupted due to distance of trace, noise, interference, etc. on the boards.

c. Electromagnetic Interference (EMI)

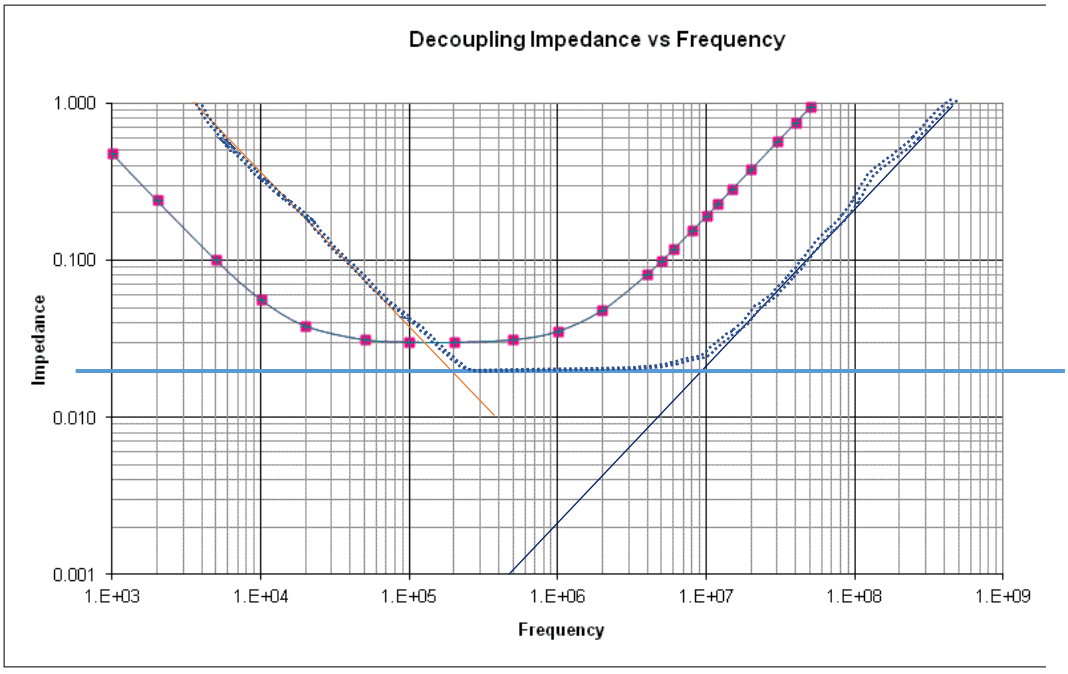
Electromagnetic Interference is any electromagnetic energy which may cause issues for an electronic device.

d. Electromagnetic Compatibility (EMC)

How well a device can both resist "expected" EMI and not generate more EMI than expected.

Q14. The following is a graph of the effective impedance of a 330µF capacitor with an ESR of 0.03Ω and an ESL of 3nH at a wide range of frequencies. Modify the curve to show what it would look like if it was replaced with a 33µF capacitor with an ESL of 300pH, and an ESR of 0.02Ω **[6]**

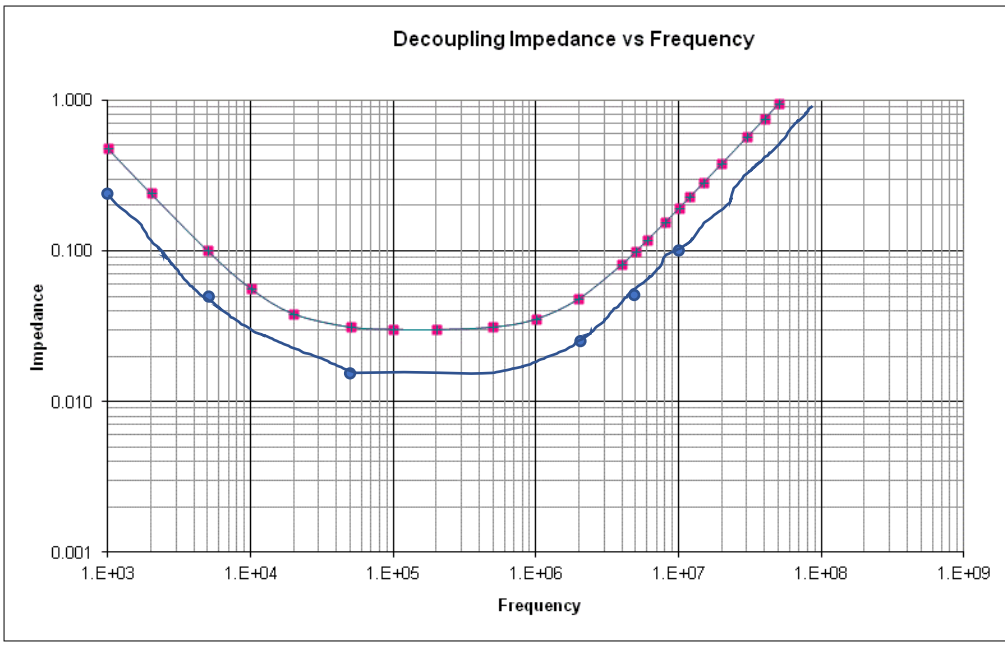
This is pretty close. I've drawn the capacitance, resistance and inductance lines and put the dotted line as the real capacitor. Not as smooth as it could be, but drawing with a mouse is tricky.



Q15. In your own words, explain why putting two capacitors in parallel is often more useful for maintaining power integrity than one capacitor of twice the size. **[5]**

Basic theme: putting two capacitors in parallel halves the ESL and ESR. A single capacitor of twice the capacitance will likely have the same or higher ESR and ESL.

Q16. This is again the graph of the effective impedance of a 330µF capacitor with an ESR of 0.03Ω and an ESL of 3nH at a wide range of frequencies. Show how the curve would change if you instead had two of those capacitors in parallel. **[5]**



Not a great drawing, but down 2 at all points.

Q17. Say you have an FPGA where the Vcc to ground value is required to be in the range of 3.27 to 3.33V. Say that FPGA can draw up to 1.5A and that the voltage regulation module generates a solid 3.3V.

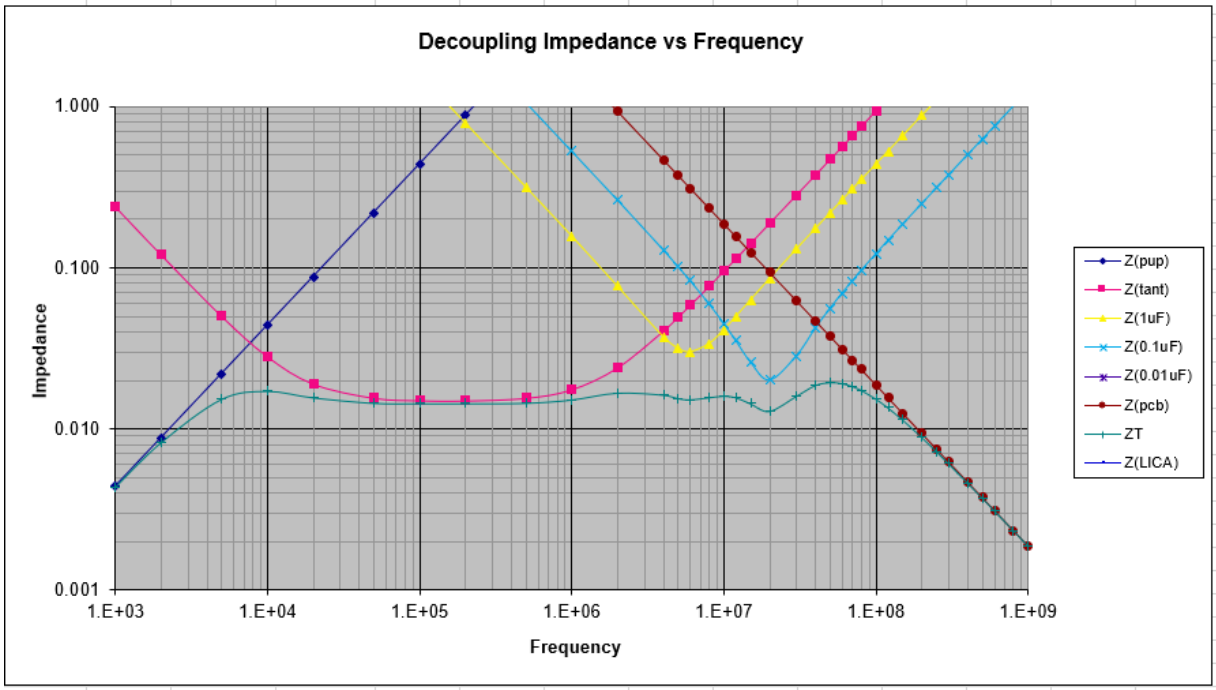
a) What is the maximum impedance your PDN (power distribution network) can have? **[3]**
V=IR; R=V/I. 0.03V/1.5A= 0.02Ohms.

b) Use the spreadsheet found on the course website with this assignment to generate a set of capacitors from those listed that will meet these requirements. You may assume you only need to worry about frequencies in the 10KHz to 900MHz range and you may include the PCB power/ground plane. Your solution should use as few capacitors as possible while meeting these requirements. **[12]**

Best I could do was 6 capacitors (plus power supply, plus PC board planes), but it's tight. We'll take some of the 7 capacitor solutions also, as I, at least, can't tell for sure that it doesn't cross the .02Ohm line.

1.8V Power supply Decoupling | John Zasio | 11/3/2003

			Each			Total	
Device	Quantity	Cap	ESR	ESL	Ceff	ESR	ESL
DC/DC Converter	1		1.00E-04	7.00E-07		1.00E-04	7.00E-07
Tantalum Capacitors	2	3.30E-04	3.00E-02	3.00E-09	6.60E-04	1.50E-02	1.50E-09
0603 Ceramic Caps.	1	1.00E-06	3.00E-02	7.00E-10	1.00E-06	3.00E-02	7.00E-10
0603 Ceramic Caps.	3	1.00E-07	6.00E-02	6.00E-10	3.00E-07	2.00E-02	2.00E-10
0603 Ceramic Caps.	0	1.00E-08	9.00E-02	5.00E-10	0.00E+00	0.00E+00	0.00E+00
0603 Ceramic Caps.	0	1.00E-09	1.50E-01	5.00E-10	0.00E+00	0.00E+00	0.00E+00
PC Board	1	8.50E-08			8.50E-08		



Q18. Say you have a processor named "Bob" which requires 3.3-4.5V and has the following run modes

Run mode	MIPS	Current
Fast	6.0	200mA
Slow	3.0	92mA
Stopped	0	10mA

If you have a task that runs once a second for 100K instructions and otherwise does nothing. It is running at room temperature. Assuming waking up and sleeping have no cost.

a. How would you use the run modes to achieve the lowest energy utilization? **[4]**
   Two choices: fast and slow. Fast would run for 1/60 of a second second at 200mA, and 59/60 of a second at 10mA= 13.17mA on average. Slow is 12.7mA on average. So slow uses less current.

b. Say you are using one of these batteries http://www.powerstream.com/thin-lithium-ion.htm (specifically the PGEB0054050[1]) to power your device. How long would you expect the power to last? (hint, think carefully about limits and assume we are directly powering the processor) **[6]**
   Well, at 92mA, it looks like it would supply somewhere around 35mAh before it fell below 3.3V. It would likely supply something close to 55mAh (the battery rating) at a 10mA draw. We don't know how the battery chemistry will treat the draw we're providing, but almost certainly it would be somewhere in that range. So, we're looking at between 35mAh/12.7mA and 55mAh/12.7mA. So about 2.75 to 4.33 hours. I'd guess something close to 4 hours.

c. What if you moved to using two of those in parallel? **[5]**
   Now you'd be drawing at most close to 1C from each battery. We don't know what that looks like exactly, but we're drawing each battery at half the rate and our worst case just got *some* better. So probably something like 6 and 9.66 hours, *probably* over 9 hours at a guess.

d. What if we put two of the batteries in series and used a linear regulator? **[5]**
   We'd get the full 55mAh out as we could make it to the battery dropping to 3.0V easily. So 4.33 hours seems darn likely.

For grading this homework, we'll be reasonably generous on answers that are close to the ranges provided. But be sure you understand these answers for the exam!

[1] It really is found on that page though it's in an image so you'll not be able to find it with a search…

Q19. Peukert Effect

a. Explain what the Peukert effect is. **[4]**

Peukert effect means that a battery has a lower capacity as the discharge current increases

b. How it is the Peukert effect reflected in the Powerstream website (from the question above)? **[6]**

The Peukert effect is reflected by the lines on the volts vs Amp-hours graph. The lines indicate that less energy is forthcoming as the current increases. Energy would be equal to the area under the curve.

Q20. Say you have a linear regulator with an 8V input and a 6V output. If the load being driven by the regulator is using 4 Watts and the quiescent current is 15mA, how much power is being wasted as heat by the regulator? Show your work. **[8]**

With 6V at 4W for the load, the current across the load is (P=IV, I=P/V) 4W/6V=0.667A or 667mA. Total power in is (667mA+15mA)*8V=5.456W. We are wasting about 1.45W

*(((4/6)+.015)*8)-4*