EECS 473 Homework 1 Fall 2026

Show your work where requested. Keep written explanations brief. Use course materials and references as needed; no hardware implementation is required. Do your own work without AI tools. Also practice an old midterm design problem when studying. Do not expect to get this back before the exam. I expect this will take about 6 hours to do. Due 10/14.

---

1. **True/False** (10 points)

Circle all TRUE statements. Start with 10 points; subtract 2 for each statement incorrectly circled or left uncircled, with a minimum of 0.

a. An LDO is a type of linear regulator.
b. Raising a linear regulator's input voltage reduces its heat, with output voltage and load current fixed.
c. A 6 cm long, 20 thou wide trace has the same resistance as a 3 cm long, 10 thou wide trace of the same thickness and temperature.
d. Gerber files are one of the outputs of the PCB CAD tool process.
e. If a set of tasks can be successfully scheduled by RM scheduling, it can also be scheduled by using EDF.
f. If you redistribute a binary licensed under the GPL to someone, you must make the source code available to them on request.
g. Software licensed under the GPL is in the public domain.
h. Capacitors resist a change in current.

2. **Scheduling algorithms** (18 points)

For Questions 2–4, assume one CPU, independent periodic tasks, deadlines equal to periods, and no scheduling overhead. RM and EDF are preemptive. Below, C is worst-case execution time and T is period, both in ms.

Group	T1 C / T	T2 C / T	T3 C / T
A	1 / 5	2 / 8	3 / 12
B	2 / 5	2 / 7	3 / 12
C	2 / 5	3 / 7	3 / 10
D	2 / 8	2 / 8	3 / 8

For each group, find CPU utilization and determine whether RM and EDF can schedule it. Use the RM utilization bound when it settles the question. Where necessary, show a critical-instant timeline sufficient to show if the group of tasks can be scheduled. In the case of equal priorities, pick one.

3. **Choosing a processor** (10 points)

Each instruction takes the same time and MIPS scales directly with execution speed. Choose among 28, 32, 40, and 48 MIPS processors.

Task	Maximum instructions per job	Period
A	0.8 million	80 ms
B	1.8 million	150 ms
C	3.0 million	300 ms

a) Find the slowest processor that can schedule these tasks under EDF. **[5]**

b) Using only utilization tests, classify each processor under RM as guaranteed schedulable, impossible, or not settled by those tests. Justify each classification; no critical-instant analysis is required. **[5]**

4. **Round Robin** (8 points)

For this problem, you are to assume an ideal scheduler: ready tasks share the CPU equally, the time quantum is negligibly small, and a task stops consuming CPU when its job finishes. All tasks first release at time 0.

a) Give two tasks, specifying execution times and periods, whose total utilization is below 75% but which miss a deadline under Round Robin. One task must have exactly 10% utilization. Show the missed deadline and explain whether EDF could schedule your pair. **[5]**

b) Give two tasks, specifying execution times and periods, whose total utilization is above 95% but which never miss a deadline under Round Robin. **[3]**

5. **Shared resources and priority inversion** (16 points)

The tasks below run on one CPU and you are to assume there is no scheduling overhead. As normal, each deadline equals its period. A and C lock mutex X before doing any work and hold it for their entire CPU execution time. B does not use X. A task waiting for X blocks rather than spins.

Task	Execution time	Period	Uses X
A	3 ms	10 ms	Yes
B	5 ms	21 ms	No
C	5 ms	33 ms	Yes

a. Ignoring X, can RM schedule these tasks? Can EDF? Briefly justify your answers. **[4]**

b. C first releases at time 0; A and B first release at time 1 ms. Draw the RM schedule through A's first deadline with no priority inheritance. Identify the priority inversion and any missed deadline. **[6]**

c. Repeat that interval with priority inheritance. Show the effective priorities while A is blocked and say whether A meets this deadline. **[4]**

d. Does priority inheritance eliminate all blocking of A? Explain what blocking remains. **[2]**

6. **Licensing** (12 points)

Read the Wikipedia article on "Gratis versus libre". Answer briefly in your own words. You may wish to consult the GNU GPL FAQ and Creative Commons license descriptions. Assume ordinary GPL-covered software with no special linking exception.

a. Explain gratis versus libre. Can software be libre without being gratis? Give an example. **[3]**

b. A company modifies a GPL program for internal use only. It later starts shipping the modified binary in its product. Explain how its source-code obligations change once it starts shipping that product and whether charging for the product is allowed. **[4]**

c. What is the practical difference between the GPL and LGPL when it comes to someone using a library in a proprietary application? **[2]**

d. Describe CC BY-NC-SA, including what ShareAlike requires when an adapted work is shared. **[3]**

7. **RTOS timing and C** (12 points)

a. Define the term "jitter" in the context of an RTOS. **[2]**

b. An interrupt-triggered operation takes exactly 0.8 ms of CPU time. Some task can disable interrupts for at most 4 ms at a time. Find the worst-case event-to-start latency, event-to-completion time, and the range of variation in completion time. Assume no other delays exist. **[3]**

c. What is a critical section? Why can protecting a long section by disabling interrupts be a problem even if it prevents a race? **[2]**

d. What is a void * in C? Why do FreeRTOS tasks take a void* as an argument? **[3]**

e. 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 compiler, not the hardware." Explain what they mean. **[2]**

8. Deferred interrupts and semaphores (12 points)

SemaphoreDeclare(BUSY) creates an empty (taken) binary semaphore; passing FREE creates an available one. SemaphoreTake blocks until available, then consumes it. SemaphoreGive makes it available; additional gives while available are ignored. Treat the `SemaphoreGive` in the ISR as the appropriate ISR-safe operation.

```
Init:
    semaphore a = SemaphoreDeclare(BUSY);
    int i = 0, j = 0;
    int x[6] = {0};

ISR:
    i++;
    SemaphoreGive(a);

Task1:
    while (1) {
        SemaphoreTake(a);
        if (i > 5) { i = 0; j++; }
        x[i] = j;
    }
```

a. The terms "top half" and "bottom half" (sometimes "upper-half" and "lower-half") are often used when discussing deferred interrupts. Explain that terminology **[2]**

b. Explain deferred interrupt processing and identify the top and bottom halves in the above code. Give one advantage of deferring the work. **[4]**

c. Why initialize **a** to BUSY? What could happen if it started FREE? **[2]**

d. Assume Task1 finishes one loop iteration after every interrupt, before the next interrupt occurs. Find x, i, and j after 22 interrupts have occurred and the corresponding 22 Task1 iterations have completed. **[4]**

9. Mutexes (8 points)

Briefly answer the following questions:

a. What is the difference between a binary semaphore and a mutex for the FreeRTOS implementation used in lab? In particular, fill in the following table with each cell having a bolded "answer" and a brief (20 words or less) explanation. Some cells have been done for you as an example. **[6]**

Feature	**Binary Semaphore**	**Mutex**
Primary Intent	**Signaling**. Task A (or an ISR) tells Task B that an event happened.	
Concept of Ownership		**Strict Ownership**. Only the specific thread that locked it can unlock it.
ISR safe?		
Priority Inheritance?		

b. What is the difference between a general (or counting) semaphore and a binary semaphore? **[2]**

10. Linux device drivers (16 points)

This function is the read member of a registered character driver for /dev/review. Assume a fresh open starts at position 0, all copies succeed, and the function is called only with a positive count. The unusual position update and return value are intentional.

```
const char s[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
ssize_t memory_read(struct file *filp, char __user *buf,
                    size_t count, loff_t *f_pos) {
    if (*f_pos >= 10)
        return 0;
    size_t x = count > 3 ? 3 : count;
    copy_to_user(buf, s + *f_pos + 1, x);
    printk(KERN_INFO "pos=%lld\n", (long long)*f_pos);
    *f_pos += x + 1;
    return x - 1;
}
```

For parts a and b, the caller repeatedly reads until read returns 0, printing exactly the number of bytes returned by each successful call.

a. The caller requests 1 byte each time. What is logged, what is printed, and why does the caller stop? **[3]**
b. After a separate fresh open, the caller requests 4 bytes each time. What is logged and printed? **[3]**
c. Rewrite the function body to return the first 10 characters of `s` contiguously across reads. Respect count and return 0 only at EOF once the 10 characters have been read. You need not include the printk. **[6]**
d. What does mknod /dev/review c 61 2 create? Explain the roles of the major and minor numbers in connecting a device node to a driver and its devices. **[4]**

11. PCB design and diagnosing noise (12 points)

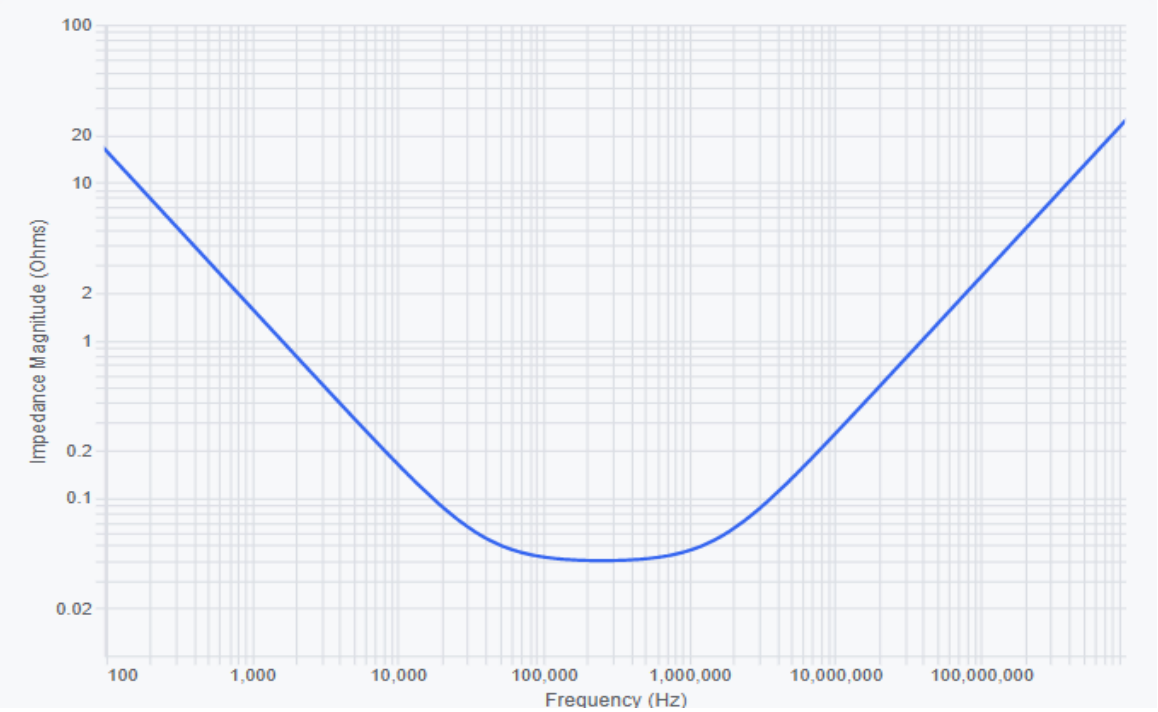
a. Briefly define PI, SI, EMI, and EMC. **[4]**
b. For each observation below, give the most likely problem category and one measurement or experiment that would help test your explanation. A symptom may have more than one plausible cause. **[6]**
    i. The processor resets when a motor starts
    ii. SPI data is occasionally wrong; MOSI rings across the input threshold even when the motor is not connected.
    iii. The LCD shows unexpected characters when a separately powered motor runs nearby.
c. Your board has a 0.1 µF capacitor connected to an IC through long, narrow traces. Why might moving the same capacitor next to the IC improve PI? **[2]**

12. Linear regulators (10 points)

a. A regulator has 7.2 V input, 5.0 V output, a 400 mA load, and 8 mA quiescent current drawn from the input. Find the input power, output power, heat dissipation, and efficiency. **[6]**
b. The same regulator supplies a 2 mA load while its quiescent current remains 8 mA. Find the efficiency and explain why quiescent current matters especially in a sleeping embedded system. **[4]**

13. Decoupling capacitors (12 points)

The graph shows a series RLC model of a 100 µF capacitor with a resistance of 0.04 Ω and an inductance of 4 nH.



a. Say we were to replace this capacitor with four identical capacitors in parallel, each with C = 25 µF, R =0.08 Ω, and L = 2 nH. Ignore interconnect parasitics. Find the equivalent C, ESR, and ESL. **[4]**
b. Sketch the new impedance curve on the same log axes (try to be close, you don't need to be exact). **[5]**
c. Why can a PCB power/ground plane help at very high frequencies despite having much less capacitance than a bulk capacitor? **[3]**

14. Processor modes and batteries (16 points)

Processor Bob requires 3.3–4.5 V. It executes 120,000 instructions every 500 ms and otherwise sleeps. Ignore wake/sleep overhead.

Mode	Speed	Current
Fast	8 MIPS	180 mA
Slow	4 MIPS	100 mA
Stopped	0 MIPS	4 mA

a. At a fixed supply voltage, compare Fast-then-Stopped with Slow-then-Stopped. Find the average current for each and choose the lower-energy strategy. **[4]**

Now use the selected strategy for the problems below. Each fully charged cell starts at 4.2 V. The following hypothetical test data apply to this exact pulsed load; no external datasheet or discharge-rate correction is needed. Cells share current equally when paralleled. Protection and balancing are already provided.

Discharge endpoint per cell	Usable charge per cell
3.3 V	90 mAh
3.0 V	100 mAh

b. With one cell connected directly, estimate operating time. Explain why using 100 mAh would be inappropriate. **[4]**
c. Repeat for two cells in parallel connected directly. **[3]**
d. Repeat for two cells in series supplying a 3.7 V linear regulator with 1 mA quiescent current and 0.2 V dropout. Discharge may continue until each cell reaches 3.0 V. Explain why series connection does not double the pack's mAh rating. **[5]**

15. **Design question** (22 points)

Consider the *answer* to the design question found on the Fall 2024 midterm. We ***strongly*** suggest you try to do the problem on your own before answering this question to get the most out of this. Where appropriate your answers should reference the handout (including a page number) and/or the problem specification. Use the page numbers of the pdf, not the page numbers of the individual documents in the pdf.

a. Part A: why are there pull up resistors on SDA and SCL? **[3]**
b. Part A: explain why HV1 is connected to A4 on the Arduino. **[4]**
c. Part B: Where is the 0x5A coming from? **[3]**
d. Part B: Why 3 bytes? **[4]**
e. Part C: Why is the VoiceBox address 7-bit number and why can't it be 0x5A? **[3]**
f. Part E: Explain the math of "tempC" and "compensated". **[5]**

16. **Project** (6 points)

Which project team are you on? Briefly, what do you think is the most likely reason your project might not be ready by the design expo? What can you do to help avoid that? What help could the 473 staff provide? Your total answer should be about 75 to 150 words.