ECE445M Exam1 Study Guide (Spring 2026):

Midterm Instructions:

- Closed book.
- Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space (boxes) provided. *Anything outside the boxes will be ignored.*
- For all questions, unless otherwise stated, find the most efficient (time, resources) solution.

This reference material will be given

```
LDR   Rd, =label    // load 32-bit address of label to Rd
LDR   Rd, [Rn]      // load 32-bit memory at [Rn] to Rd
STR   Rt, [Rn]      // store Rt to 32-bit memory at [Rn]
LDR   Rd, [Rn, #n]  // load 32-bit memory at [Rn+n] to Rd
STR   Rt, [Rn, #n]  // store Rt to 32-bit memory at [Rn+n]
MOV   Rd, Rn        // Rd = Rn
MOVS  Rd, #imm8     // Rd = M
ADDS  Rd, Rn, Rm    // Rd = Rn + Rm
ADDS  Rd, Rn, #imm8 // Rd = Rn + M
SUBS  Rd, Rn, Rm    // Rd = Rn - Rm
SUBS  Rd, Rn, #imm8 // Rd = Rn - M
CPSID I             // disable interrupts, I=1
CPSIE I             // enable interrupts, I=0
B     label         // branch to label
BX    Rm            // branch indirect to location specified by Rm
BL    label         // branch to subroutine at label
SVC   #imm8         // invoke the SVC_Handler
PUSH  {Rn,Rm}       // push Rn and Rm onto the stack
PUSH  {Rn-Rm}       // push all registers from Rn to Rm onto stack
POP   {Rn,Rm}       // pop two 32-bit numbers off stack into Rn, Rm
POP   {Rn-Rm}       // pop multiple 32-bit off stack to Rn - Rm
```

Lecture notes: Lectures 1-4

Book: Chapters 1-6 (except 3)

1. Computer Architecture

1.1. Introduction to Real-Time Operating Systems
1.2. Computer Architecture
1.3. Cortex-M Processor Architecture
~~1.4. Texas Instruments Cortex-M Microcontrollers~~
1.5. ARM Cortex-M Assembly Language
1.6. Pointers in C
1.7. Linking Assembly to C
1.8. Macros in C
1.9. Function Pointers in C

2. Microcontroller Input/Output

2.1. Parallel I/O (concepts only, no MSPM0 details)
2.2. Interrupts (concepts only, no MSPM0 details)
2.4. Edge-triggered Interrupts (concepts only, no MSPM0 details)

2.5. UART Interface (concepts only, no MSPM0 details)
~~2.6. Synchronous Transmission and Receiving using the SSI~~
~~2.7. Pulse Width Modulation~~
~~2.9. Analog Output~~
2.10. Analog Input (concepts only, no MSPM0 details)
2.11. Introduction to Debugging

~~**3. High Speed Input/Output**~~

4. Thread Management
4.1. Introduction to RTOS
4.2. Thread Management
4.3. Periodic Tasks
4.4. Thread Suspend
4.5. Thread Sleeping
4.6. Thread Creation and Killing

5. Thread Communication/Synchronization
5.1. Spin-lock Semaphores
5.2. Spin-lock Semaphores with Cooperation
5.2. Blocking semaphores
5.3. Thread Synchronization
5.4. Producer-Consumer Problems
5.5. Debouncing a switch (concepts only, no MSPM0 details)
~~5.6. Monitors~~
~~5.7. Path Expressions~~
5.8. Deadlocks

6. Thread Scheduling
6.1. Introduction to Scheduling
6.2. Cooperative Scheduler
~~6.3. Priority scheduler~~
~~6.4. Fixed Scheduling~~
~~6.5. Other Scheduling Algorithms~~

Labs: Labs 1-2, important topics
Lab 1: Interrupts, Cortex M architecture, FIFO queues, UART, ADC
Lab 2: Real time OS, semaphores, critical sections, synchronization, communication

Architecture (concepts only, no MSPM0 details)
Registers, buses, ports, stack,
Interrupts NVIC, tail chain, late arriving,
SysTick, PendSV, edge triggered interrupts,
arm, enable, latency, jitter, hardware FIFO,
ARM assembly code,

subroutine linkage (AAPCS),
parameter passing (registers and stack),
local variables (registers and stack),
interrupt linkage.

Data flow graph, call graph, flowcharts.

HAL, device driver.

Debugging

intrusiveness,
profile,
dump,
control, observability,
coverage
white box, black box

Data structures

FIFO,
statically allocated linked lists,
dynamically allocated linked lists,

OS stuff

latency, real- time, interrupt priority,
reentrancy, critical sections, race condition,
sleeping,
scheduling
preemptive vs nonpreemptive=cooperative,
round robin scheduler,
~~priority scheduler,~~
~~rate monotonic,~~
~~earliest deadline first,~~
~~least slack-time first,~~
semaphore implementation
spinlock,
cooperative spinlock,
~~blocking,~~
semaphore applications
deadlocks
necessary conditions,
detection,
prevention,
avoidance
OS concepts
kernel,
hard/soft/firm real time
~~bounded waiting,~~
~~priority inversion, priority inheritance, aging, starvation,~~

mutual exclusion,
certification,
hooks,
slack time,
lateness,
~~rate monotonic scheduler, rate monotonic theorem,~~
protection (using both SPs),
CPU utilization,
Deadlock: prevention, detection, resource allocation graph,
~~Monitors~~.

See questions at the end of Chapter 4), Chapter 5) ~~and Chapter 6).~~

Past exams:

Real time OS, semaphores, critical sections, synchronization, communication
Spring 2001,Quiz2, Question 2, Sleep primitive
~~Fall 2001,Quiz2, Question 4, Priority scheduler, deadlock~~
Spring 2002, Quiz1, Question 3, Dynamic thread allocation, thread Kill
Fall 2002, Quiz2, Question 2, application of semaphores
Fall 2002, Final, Question 4, use of semaphores
Fall 2002, Final, Bonus questions 1,2,6, assembly language used in OS programming
Fall 2003, Quiz1, Question 2, use of semaphores
Fall 2003, Quiz1, Question 3, changing the TCB
Fall 2003, Quiz1, Question 4, definition of time jitter
Fall 2003, Quiz1, Question 5, implementation of OS_Wait
Fall 2003, Final, Question 14, definitions of OS concepts/terms
Fall 2004, Quiz2, Question 2, Three thread rendezvous
Fall 2004, Quiz2, Question 3, Binary semaphore
~~Fall 2004, Final, Question 9, Path expression~~
Fall 2005, Quiz2, Question 4, Reader/writer problem
Fall 2005, Quiz2, Question 5, Cooperative thread scheduler
Fall 2006, Quiz2, Question 9, Fork
Fall 2006, Quiz2, Question 5, Resource allocation graph
~~Fall 2006, Final, Question 5, Exponential Queue or multi-level feedback queue scheduling~~
Spring 2008, Quiz2, Question 4, use of semaphores
Spring 2008, Final, Question 2, Effect of OS on time-jitter while sampling an ADC
Spring 2008, Final, Question 5, Critical section, design new instruction
Spring 2009, Quiz 2, Question 4, Critical section
Spring 2009, Quiz 2, Question 5, Fork and join
Spring 2009, Final, Question 5, kill threads that finish executing
Spring 2010, Quiz 1, Question 2, word bank
Spring 2010, Quiz 1, Question 4, alternate words for signal and wait
Spring 2010, Quiz 1, Question 5, what happens if an ISR calls OS_Wait
Spring 2010, Quiz 1, Question 6, implementing mutual exclusion
Spring 2010, Quiz 1, Question 7, application of semaphores
Spring 2011, Quiz 1, Question 4, definitions

Spring 2011, Quiz 1, Question 5, application of semaphores
Spring 2011, Quiz 1, Question 6, new implementation of semaphores
~~Spring 2011, Quiz 1, Question 7, priority scheduler (the 2011 class did horrible on this question because they parroted their lab solution without reading the question)~~
Spring 2010 Final, Question 6, definitions i, j
~~Spring 2011 Final, Question 8, bounded waiting~~
Spring 2011 Final, Question 9, real time OS, minimizing latency
Spring 2011 Final, Question 11, FIFO with semaphores
Spring 2011 Final, Question 12, implementing semaphores in a Dual core processor
Spring 2011 Final, Question 16, implementing a thread scheduler one a 16-core processor
Spring 2012 Quiz 1, Question 4, Two SPs.
Spring 2012 Quiz 1, Question 5 b, d, e, OS definitions.
~~Spring 2012 Quiz 1, Question 7, Monitor and deadlocks.~~
Spring 2012 Quiz 1, Question 8, OS_AddThread and OS_Kill.
Spring 2012 Quiz 1, Question 9, Use OS to debounce a switch.
Spring 2013 Quiz 1, Question 1, Priority.
Spring 2013 Quiz 1, Question 3, OS definitions.
Spring 2013 Quiz 1, Question 5, using semaphores.
Spring 2013 Quiz 1, Question 6, Assembly language thread switch.
Spring 2015 Midterm, Problem 3, weighted round robin
Spring 2015 Midterm, Problem 5, dining philosopher with semaphores
Spring 2015 Midterm, Problem 6, rendezvous/barrier with semaphores
Spring 2016 Midterm, Problem 1, critical sections and deadlock
~~Spring 2016 Midterm, Problem 2, priority scheduler~~
Spring 2016 Midterm, Problem 3, OS_Sleep
Spring 2016 Midterm, Problem 4, Kill thread on main thread exit
Spring 2016 Midterm, Problem 5a) b), synchronization and deadlock
Spring 2017 Quiz 1, Question 2, Hard fault in the OS.
Spring 2017 Quiz 1, Question 4, Detecting deadlocks in semaphores.
Spring 2017 Quiz 1, Question 5, Context switch in SysTick assembly.
Spring 2017 Quiz 1, Question 6, Bug using semaphores.
Spring 2018 Midterm, Problem 1 a) b), critical sections and deadlock
Spring 2018 Midterm, Problem 2, sequence of operation
Spring 2018 Midterm, Problem 3, periodic tasks
Spring 2018 Midterm, Problem 4, spin-lock semaphore
Spring 2019 Midterm, Problem 1, context switch in assembly
Spring 2019 Midterm, Problem 3, predecessor/successor synchronization with semaphores
Spring 2019 Midterm, Problem 4, guess what scheduler is being used
Fall 2020 Midterm, Problem 1, semaphore implementation (OR condition)
Fall 2020 Midterm, Problem 2, debugging dump
Fall 2020 Midterm, Problem 3, race conditions and reentrancy
Fall 2020 Midterm, Problem 4, background/event
Fall 2020 Midterm, Problem 6 a) b), scheduler
Spring 2022 Midterm, Problem 1, PendSV handler in assembly
Spring 2022 Midterm, Problem 2, Relationship between SysTick and PendSV handlers
Spring 2022 Midterm, Problem 4, Synchronization and deadlocks

Spring 2022 Midterm, Problem 5, Bulk Synchronization with semaphores
Spring 2023 Midterm, Problem 1, OS concepts
Spring 2023 Midterm, Problem 2, PendSV handler in assembly
Spring 2023 Midterm, Problem 3, Synchronization with semaphores
Spring 2023 Midterm, Problem 3, Semaphore with timeout
Spring 2024 Midterm, Problem 1 a) b) d), OS concepts
Spring 2024 Midterm, Problem 3, Synchronization with semaphores
Spring 2024 Midterm, Problem 4, Synchronization with semaphores

General questions
Fall 2004, Quiz2, Question 4, Time-jitter
Fall 2004, Quiz2, Question 5, Definitions and a word bank
Fall 2005, Quiz2, Question 6, Time-jitter
Fall 2006, Final, Question 4, Critical section
Spring 2009, Quiz 2, Question 3, FIFO implementation
Spring 2011, Quiz 1, Question 1, time jitter
Spring 2011, Quiz 1, Question 2, reentrant, parameter passing, LR
Spring 2011, Quiz 1, Question 3, bit-banded I/O eliminates critical section, which registers are pushed on the stack during an interrupt context switch, what is LR during an ISR
Spring 2010 Final, Question 1, Cortex M3 interrupt context switch (answer for MSPM0)
Spring 2011 Final, Question 2, Cortex M3 interrupt context switch
Spring 2012 Quiz 1, Question 3, Harvard architecture.
Spring 2012 Quiz 1, Question 6, Reentrancy.
Spring 2013 Quiz 1, Question 2, Control and observability.
Spring 2013 Quiz 1, Question 4, Critical section
Spring 2015 Midterm, Problem 1, Reentrancy
Spring 2015 Midterm, Problem 2, stack size, Reentrancy
Spring 2015 Midterm, Problem 4, real time events
Spring 2017 Quiz 1, Question 1, Critical sections with three ISAs.
Spring 2019 Midterm, Problem 2, critical sections