MIT 6.1810 Operating Systems - Study Plan for Working Professionals

Course Overview

- **Course**: MIT 6.1810: Operating System Engineering
- **Website**: https://pdos.csail.mit.edu/6.1810/2023/overview.html
- **Start Date**: August 4, 2025
- **Duration**: 12-15 months (flexible timeline)
- **Estimated Completion**: August 2026 - November 2026
- **Time Commitment**: 2 hours/week (104 hours total)
- **Target**: Software Systems Architect with embedded systems background
- **Focus**: RISC-V architecture with xv6 operating system

Prerequisites Met

- Software Systems Architecture background
- Embedded systems experience
- Basic Operating Systems concepts
- **Learning Goal**: RISC-V specific OS implementation

Study Plan Timeline with Concrete Dates

Week	Start Date	End Date	Topic	Lab/Activity	Weekly Objectives
1	**Aug 4, 2025**	**Aug 10, 2025**	Course Setup & Environment	Lab 0: Environment Setup	1. Set up xv6 development environment 2. Compile and run xv6 3. Understand basic xv6 structure
2	**Aug 11, 2025**	**Aug 17, 2025**	Unix Utilities Foundation	Lab 1: Unix Utilities (sleep, pingpong)	1. Implement sleep utility 2. Implement pingpong utility 3. Understand system calls and process creation
3	**Aug 18, 2025**	**Aug 24, 2025**	Unix Utilities Advanced	Lab 1: Unix Utilities (primes, find, xargs)	1. Complete primes utility 2. Implement find and xargs utilities 3. Master pipes and process communication
4	**Aug 25, 2025**	**Aug 31, 2025**	System Calls Introduction	Lab 2: System Calls (trace)	1. Implement system call tracing 2. Understand kernel-user interface 3. Learn system call mechanism

Week	Start Date	End Date	Topic	Lab/Activity	Weekly Objectives
5	**Sep 1, 2025**	**Sep 7, 2025**	System Calls Deep Dive	Lab 2: System Calls (sysinfo)	1. Complete sysinfo system call 2. Understand kernel data structures 3. Master system call implementation
6	**Sep 8, 2025**	**Sep 14, 2025**	RISC-V Architecture Study	Reading: xv6 book Ch 1-2, RISC-V primer	1. Learn RISC-V ISA fundamentals 2. Understand registers and instruction formats 3. Study RISC-V assembly basics
7	**Sep 15, 2025**	**Sep 21, 2025**	RISC-V Memory Model	Reading: RISC-V memory model, privilege levels	1. Understand RISC-V memory model 2. Learn supervisor/user modes 3. Study Control and Status Registers (CSRs)
8	**Sep 22, 2025**	**Sep 28, 2025**	Virtual Memory Concepts	Reading: xv6 book Ch 3, VM theory	1. Study virtual memory concepts 2. Understand address translation 3. Learn page table fundamentals
9	**Sep 29, 2025**	**Oct 5, 2025**	Page Tables Implementation	Lab 3: Page Tables (speed up system calls)	1. Implement page table optimizations 2. Understand MMU operations 3. Optimize system call performance
10	**Oct 6, 2025**	**Oct 12, 2025**	Page Table Management	Lab 3: Page Tables (detect accessed pages, print)	1. Complete page table lab 2. Implement page access detection 3. Master virtual memory manipulation
11	**Oct 13, 2025**	**Oct 19, 2025**	Memory Management Theory	Reading: MMU, TLB, address translation	1. Understand hardware memory management 2. Learn TLB operations 3. Study address translation mechanisms
12	**Oct 20, 2025**	**Oct 26, 2025**	Traps and Interrupts	Lab 4: Traps (backtrace)	1. Implement stack backtrace 2. Understand trap mechanism 3. Learn interrupt handling basics
13	**Oct 27, 2025**	**Nov 2, 2025**	Timer Interrupts	Lab 4: Traps (alarm)	1. Implement alarm system call 2. Master timer interrupt handling 3. Understand periodic interrupts
14	**Nov 3, 2025**	**Nov 9,**	Interrupt Deep Dive	Reading: xv6 book Ch 4, RISC-V interrupt spec	1. Study interrupt controllers

Week	Start Date	End Date	Topic	Lab/Activity	Weekly Objectives
		2025			2. Learn exception handling mechanisms 3. Understand interrupt priorities
15	**Nov 10, 2025**	**Nov 16, 2025**	Exception Handling	Reading: RISC-V exceptions, trap vectors	1. Understand exception types 2. Learn trap vector setup 3. Study context switching
16	**Nov 17, 2025**	**Nov 23, 2025**	Copy-on-Write Concepts	Lab 5: Copy-on-Write (lazy allocation)	1. Implement lazy memory allocation 2. Understand COW principles 3. Learn memory optimization techniques
17	**Nov 24, 2025**	**Nov 30, 2025**	COW Fork Implementation	Lab 5: Copy-on-Write (COW fork)	1. Complete COW fork optimization 2. Master memory sharing 3. Understand process memory management
18	**Dec 1, 2025**	**Dec 7, 2025**	Memory Optimization Study	Reading: Advanced memory management papers	1. Study modern memory management techniques 2. Learn about garbage collection 3. Understand memory allocation strategies
19	**Dec 8, 2025**	**Dec 14, 2025**	Memory Allocation Strategies	Reading: Malloc implementations, memory pools	1. Understand memory allocation algorithms 2. Learn about fragmentation issues 3. Study memory pool techniques
20	**Dec 15, 2025**	**Dec 21, 2025**	Threading Fundamentals	Lab 6: Multithreading (user threads)	1. Implement user-level threading 2. Understand thread switching 3. Learn thread lifecycle management
21	**Dec 22, 2025**	**Dec 28, 2025**	Synchronization Primitives	Lab 6: Multithreading (barriers, locks)	1. Implement barriers and locks 2. Master synchronization 3. Understand thread coordination
22	**Dec 29, 2025**	**Jan 4, 2026**	Concurrency Theory	Reading: xv6 book Ch 6, race conditions	1. Study race conditions 2. Learn about deadlocks 3. Understand synchronization patterns
23	**Jan 5, 2026**	**Jan 11, 2026**	Advanced Concurrency	Reading: Lock-free programming, atomic operations	1. Understand lock-free data structures 2. Learn memory ordering 3. Study atomic operations
24	**Jan 12, 2026**	**Jan 18, 2026**	Network Driver Basics	Lab 7: Network Driver (E1000 setup)	1. Set up E1000 driver framework 2. Understand PCI devices

Week	Start Date	End Date	Topic	Lab/Activity	Weekly Objectives
					3. Learn device initialization
25	**Jan 19, 2026**	**Jan 25, 2026**	Network I/O Implementation	Lab 7: Network Driver (transmit/receive)	1. Implement packet transmission 2. Implement packet reception 3. Understand DMA operations
26	**Jan 26, 2026**	**Feb 1, 2026**	Device Driver Patterns	Reading: Device driver architecture, I/O models	1. Study device driver patterns 2. Learn interrupt-driven I/O 3. Understand driver architecture
27	**Feb 2, 2026**	**Feb 8, 2026**	Memory-Mapped I/O	Reading: MMIO, DMA, bus architectures	1. Understand memory-mapped I/O 2. Learn bus communication protocols 3. Study DMA mechanisms
28	**Feb 9, 2026**	**Feb 15, 2026**	Lock Performance	Lab 8: Locks (fine-grained locking)	1. Implement fine-grained locking 2. Reduce lock contention 3. Optimize lock performance
29	**Feb 16, 2026**	**Feb 22, 2026**	Lock-Free Structures	Lab 8: Locks (lock-free data structures)	1. Implement lock-free data structures 2. Understand atomic operations 3. Learn lockless programming
30	**Feb 23, 2026**	**Mar 1, 2026**	Performance Analysis	Profiling tools, performance measurement	1. Learn performance profiling 2. Identify bottlenecks 3. Use profiling tools
31	**Mar 2, 2026**	**Mar 8, 2026**	System Optimization	Performance tuning, cache optimization	1. Apply optimization techniques 2. Understand cache behavior 3. Optimize system performance
32	**Mar 9, 2026**	**Mar 15, 2026**	File System Basics	Lab 9: File Systems (large files)	1. Implement large file support 2. Understand inode structure 3. Learn file system organization
33	**Mar 16, 2026**	**Mar 22, 2026**	Symbolic Links	Lab 9: File Systems (symbolic links)	1. Implement symbolic links 2. Understand directory operations 3. Learn path resolution
34	**Mar 23,**	**Mar 29,**	Storage Systems Study	Reading: File system papers, B-trees	1. Study modern file systems

Week	Start Date	End Date	Topic	Lab/Activity	Weekly Objectives
	2026	**2026**			2. Learn indexing structures 3. Understand B-tree operations
35	**Mar 30, 2026**	**Apr 5, 2026**	Storage Hierarchy	Reading: Storage technologies, caching	1. Understand storage hierarchy 2. Learn disk scheduling algorithms 3. Study caching strategies
36	**Apr 6, 2026**	**Apr 12, 2026**	Memory Mapping Basics	Lab 10: mmap (memory-mapped files)	1. Implement memory-mapped files 2. Understand virtual memory integration 3. Learn mmap semantics
37	**Apr 13, 2026**	**Apr 19, 2026**	Lazy Loading	Lab 10: mmap (lazy loading)	1. Implement lazy loading for mmap 2. Optimize memory usage 3. Understand demand paging
38	**Apr 20, 2026**	**Apr 26, 2026**	Advanced VM Topics	Reading: Advanced VM papers, paging algorithms	1. Study advanced virtual memory systems 2. Learn page replacement algorithms 3. Understand VM optimization
39	**Apr 27, 2026**	**May 3, 2026**	VM Integration	Reading: VM and file system integration	1. Understand VM-filesystem interaction 2. Learn unified buffer cache 3. Study memory-file integration
40	**May 4, 2026**	**May 10, 2026**	System Integration	System integration, comprehensive testing	1. Integrate all components 2. Ensure system stability 3. Test complete system
41	**May 11, 2026**	**May 17, 2026**	Testing and Debugging	Comprehensive testing, bug fixes	1. Test complete system 2. Fix integration issues 3. Ensure system reliability
42	**May 18, 2026**	**May 24, 2026**	Optional Challenges (Part 1)	Choose 2-3 optional challenges based on interest	1. Deepen understanding in areas of interest 2. Implement advanced features 3. Explore specialized topics
43	**May 25, 2026**	**May 31, 2026**	Optional Challenges (Part 2)	Continue optional challenges	1. Explore advanced topics 2. Implement additional features 3. Master specialized areas
44	**Jun 1, 2026**	**Jun 7, 2026**	Advanced Topics	Complete remaining optional challenges	1. Master advanced concepts 2. Prepare for next steps

Week	Start Date	End Date	Topic	Lab/Activity	Weekly Objectives
					3. Complete specialized implementations
45	**Jun 8, 2026**	**Jun 14, 2026**	Performance Optimization	Optimize implementations, measure improvements	1. Apply performance optimizations 2. Benchmark results 3. Measure system improvements
46	**Jun 15, 2026**	**Jun 21, 2026**	Security Enhancements	Implement security features, vulnerability analysis	1. Add security features 2. Understand OS security principles 3. Analyze vulnerabilities
47	**Jun 22, 2026**	**Jun 28, 2026**	Documentation and Review	Document implementations, create technical summaries	1. Document learning 2. Prepare technical presentations 3. Create comprehensive summaries
48	**Jun 29, 2026**	**Aug 2, 2026**	Final Integration	Final testing, complete remaining work	1. Complete all outstanding work 2. Prepare for advanced topics 3. Finalize implementations
49-52+	**Aug 3, 2026**	**Nov 1, 2026**	**Buffer/Catch-up**	Complete any remaining work, review, prepare for advanced topics	1. Ensure solid foundation before moving to advanced OS topics 2. Complete any remaining labs 3. Prepare for next learning phase

Lab Details and Priorities

Core Labs (Must Complete)

1. **Lab 1: Unix Utilities** - Foundation for system calls
   - `sleep`, `pingpong`, `primes`, `find`, `xargs`
   - **Time**: 4 hours
   - **Key Skills**: System calls, process creation, pipes
2. **Lab 2: System Calls** - Kernel-user interface
   - `trace`, `sysinfo`
   - **Time**: 4 hours
   - **Key Skills**: Adding system calls, kernel data structures
3. **Lab 4: Traps** - Interrupt and exception handling
   - `backtrace`, `alarm`
   - **Time**: 4 hours
   - **Key Skills**: RISC-V trap handling, timer interrupts
4. **Lab 6: Multithreading** - Concurrency fundamentals

- User-level threads, barriers, locks
- **Time**: 4 hours
- **Key Skills**: Thread switching, synchronization

5. **Lab 9: File Systems** - Storage and persistence
   - Large files, symbolic links
   - **Time**: 4 hours
   - **Key Skills**: Inode structure, directory operations

High Priority Labs

6. **Lab 3: Page Tables** - Virtual memory management
   - Speed up system calls, detect accessed pages
   - **Time**: 4 hours
   - **Key Skills**: Page table manipulation, virtual memory
7. **Lab 5: Copy-on-Write** - Memory optimization
   - COW fork, lazy allocation
   - **Time**: 4 hours
   - **Key Skills**: Memory management optimization
8. **Lab 10: mmap** - Memory-mapped I/O
   - Memory-mapped files, lazy loading
   - **Time**: 4 hours
   - **Key Skills**: Virtual memory integration

Lower Priority Labs (Can be simplified if needed)

9. **Lab 7: Network Driver** - Device drivers
   - E1000 network driver
   - **Time**: 4 hours
   - **Key Skills**: Device driver development, interrupts
10. **Lab 8: Locks** - Performance optimization
    - Fine-grained locking, lock-free data structures
    - **Time**: 4 hours
    - **Key Skills**: Lock optimization, performance tuning

Study Strategy for Working Professionals

Weekly Schedule

- **Weekdays (30-45 min sessions)**:
  - Reading assignments
  - Code review and planning
  - Small coding tasks
  - Documentation and notes

- **Weekends (1-1.5 hr sessions)**:
  - Major implementation work
  - Debugging and testing
  - Lab completion
  - Monthly reviews

Leveraging Your Background

Embedded Systems Experience

- ✅ Hardware-software interfaces (advantage in device drivers)
- ✅ Memory-constrained programming (helpful for xv6 limitations)
- ✅ Real-time concepts (applicable to interrupt handling)
- 📈 **Focus**: RISC-V specific details vs ARM/x86 experience

Systems Architecture Knowledge

- ✅ Understanding of layered systems (OS abstraction levels)
- ✅ Performance considerations (cache, memory hierarchy)
- ✅ Concurrency patterns (thread management, synchronization)
- 📈 **Focus**: Academic implementation vs production systems

Monthly Review Structure

Week 1: Technical Review

- What new concepts did I learn?
- What implementation challenges did I face?
- How do xv6 solutions compare to production systems?

Week 2: Connection to Professional Experience

- How does this relate to my embedded systems work?
- What patterns can I apply in my current role?
- What would I do differently in a production environment?

Week 3: Knowledge Consolidation

- Can I explain the concepts to a colleague?
- What are the key takeaways for system design?
- Where are the gaps in my understanding?

Week 4: Planning and Adjustment

- Am I on track with the timeline?
- Do I need to adjust the pace?
- What should I focus on next month?

Resources and References

Primary Resources

- **xv6 Book**: https://pdos.csail.mit.edu/6.828/2023/xv6/book-riscv-rev3.pdf
- **Course Website**: https://pdos.csail.mit.edu/6.1810/2023/overview.html
- **Lab Instructions**: https://pdos.csail.mit.edu/6.1810/2023/labs/

RISC-V Resources

- **RISC-V ISA Specification**: https://riscv.org/specifications/
- **RISC-V Assembly Guide**: https://github.com/riscv-non-isa/riscv-asm-manual
- **RISC-V Privileged Architecture**: Focus on supervisor mode

Development Environment

- **Local Setup**: Use your existing Linux development environment
- **Alternative**: Docker container for consistent environment
- **Debugging**: GDB with RISC-V support
- **Editor**: VS Code with C/C++ extensions

Supplementary Reading

- **Operating System Concepts** (Silberschatz) - for theoretical background
- **Computer Systems: A Programmer's Perspective** (Bryant & O'Hallaron) - for systems programming
- **The Design and Implementation of the FreeBSD Operating System** - for production OS comparison

Optional Challenges by Interest Area

Performance Optimization

- Lab 1: Shell improvements with history and tab completion
- Lab 5: Measure COW performance improvements
- Lab 8: Lock-free data structures implementation

Security and Reliability

- Lab 3: Null pointer dereference detection
- Lab 4: Enhanced backtrace with function names
- Lab 6: Thread safety improvements

Networking and I/O

- Lab 7: Full ARP cache implementation
- Lab 7: Multiple RX/TX rings
- Lab 7: TCP stack implementation

File Systems and Storage

- Lab 9: Triple-indirect blocks
- Lab 10: Page-out and page-in implementation
- Lab 10: Shared memory for mmap

Timeline Flexibility

If Ahead of Schedule

- Dive deeper into optional challenges
- Study related research papers
- Implement additional features
- Prepare for advanced topics (distributed systems, etc.)

If Behind Schedule

- Focus on core labs (1, 2, 4, 6, 9)
- Simplify implementations (basic requirements only)
- Skip optional challenges initially
- Use buffer months (13-15) for catch-up

Adjustment Points

- **Month 3 (Oct 26, 2025)**: Assess progress after first complex lab (page tables)
- **Month 6 (Jan 18, 2026)**: Mid-course evaluation and timeline adjustment
- **Month 9 (Apr 12, 2026)**: Final push decision - depth vs breadth
- **Month 12 (Aug 2, 2026)**: Completion assessment and next steps planning

Success Metrics

Technical Milestones

- [ ] All core labs completed and tested
- [ ] Understanding of RISC-V architecture
- [ ] Ability to explain OS concepts clearly
- [ ] Integration of all components working

Professional Development

- [ ] Enhanced systems programming skills
- [ ] Deeper understanding of hardware-software interface
- [ ] Improved debugging and problem-solving abilities
- [ ] Foundation for advanced OS topics

Next Steps Preparation

- [ ] Ready for distributed systems course (6.824)
- [ ] Prepared for systems research projects
- [ ] Enhanced capability for system-level architecture decisions
- [ ] Solid foundation for OS kernel development

Notes and Progress Tracking

Lab Completion Checklist

- [ ] Lab 1: Unix Utilities
- [ ] Lab 2: System Calls
- [ ] Lab 3: Page Tables
- [ ] Lab 4: Traps
- [ ] Lab 5: Copy-on-Write
- [ ] Lab 6: Multithreading
- [ ] Lab 7: Network Driver
- [ ] Lab 8: Locks
- [ ] Lab 9: File Systems
- [ ] Lab 10: mmap

Monthly Progress Log

```
Month 1 (Aug 4-31, 2025): [Progress] - [Notes]
Month 2 (Sep 1-28, 2025): [Progress] - [Notes]
Month 3 (Sep 29-Oct 26, 2025): [Progress] - [Notes]
Month 4 (Oct 27-Nov 23, 2025): [Progress] - [Notes]
Month 5 (Nov 24-Dec 21, 2025): [Progress] - [Notes]
Month 6 (Dec 22, 2025-Jan 18, 2026): [Progress] - [Notes]
Month 7 (Jan 19-Feb 15, 2026): [Progress] - [Notes]
```

```
Month 8 (Feb 16-Mar 15, 2026): [Progress] - [Notes]
Month 9 (Mar 16-Apr 12, 2026): [Progress] - [Notes]
Month 10 (Apr 13-May 10, 2026): [Progress] - [Notes]
Month 11 (May 11-Jun 7, 2026): [Progress] - [Notes]
Month 12 (Jun 8-Aug 2, 2026): [Progress] - [Notes]
Buffer Period (Aug 3-Nov 1, 2026): [Progress] - [Notes]
```

Key Insights and Learnings

```
[Date] - [Topic] - [Insight] - [Application to work]
```

Remember: This is a marathon, not a sprint. Consistency over intensity. Your professional experience is an asset - use it to understand the "why" behind implementations, not just the "how".

Good luck with your OS journey! 🚀