

Department of Electrical and Computer Engineering  
University of Wisconsin-Madison

**ECE 353**  
**INTRODUCTION TO MICROPROCESSOR**  
**SYSTEMS**

Educational Objectives

Spring 2008

Michael G. Morrow

Last updated 1/20/2008 11:16 PM

COURSE OBJECTIVES

---

**Module 1**

Introduction to Microprocessor Systems

**Readings**

Ch. 1

- [1] Identify the components of a microprocessor system.
- [2] Explain how embedded systems typically differ from general-purpose computer systems.
- [2] Describe the basic structure and functions of a microprocessor system.
- [2] Describe the purpose of the address, data, and control lines in microprocessor memory system.
- [2] Describe the operation of active-high and active-low signals.
- [3] Draw the memory map for a given system configuration.
- [2] Identify the design decisions available to implement an embedded microprocessor system.

Microprocessor Organization

Ch. 2-3  
AARM Preface, 1  
ARM7 1

- [2] Describe the elements of a system that are contained in programmer's model.
- [4] Compare and contrast CISC and RISC architectures.
- [4] Compare and contrast memory architectures used in microprocessor systems.
- [4] Compare and contrast memory-mapped I/O with isolated I/O.
- [3] Use the proper prefixes to calculate the size of memory in bit, byte, and word units.
- [3] For a given data size and endianness, determine the actual values stored in each byte of memory.
- [2] Describe the basic organization of the ARM microprocessors.
- [2] Describe the significance of the designation ARM7TDMI.
- [2] Describe the basic organization of the ADuC7026 microprocessor.

**Module 2**

Development Tools, ARM7TDMI Organization

**Readings**Ch. 5, 6.1-6.11, 6.14  
ARM7 2,  
Supp #1

- [2] Describe the general syntax of an assembly language instruction.
- [2] Describe the code generation process, including assembly, linking, locating, and hex conversion.
- [2] Describe the purpose and content of listing, map, and hex files.
- [4] Compare the relative strengths and weaknesses of common debugging tools.
- [2] Describe the various phases/activities of a typical embedded system development process.
- [2] Describe the characteristics of high-level languages, assembly language, machine language, native assemblers, and cross-assemblers.
- [2] Describe the registers that make up the ARM7TDMI programmer's model and any special functions that registers have.
- [2] Describe the operating modes of the ARM7TDMI, including how they are entered and what they are intended for.
- [2] Describe how register banking is used in the ARM7TDMI.
- [3] Given an instruction and operands, determine the encoded instruction's value.
- [3] Given an opcode, determine the corresponding assembly language instruction and operands.
- [2] Describe how updates to flags in ARM7TDMI CPSR register are made, and what each flag bit represents.
- [4] Compare and contrast segmented and linear memory architectures.

Addressing Modes, ARM7TDMI Instruction Set

Ch. 4,  
ARMINSTREF

- [2] Describe the addressing modes commonly supported in microprocessors.
- [3] Interpret register transfer expressions to determine the outcome of an instruction's execution.
- [3] Use the EQU directive to declare constants.
- [3] Determine the appropriate ARM7TDMI addressing modes to use in accessing registers, memory, and I/O devices.
- [2] Describe the special addressing mode cases when R15 is the destination operand.
- [3] Write assembly language instructions using all ARM7TDMI addressing modes.
- [2] Describe the purpose of the ENTRY, AREA, ARM, THUMB, EXPORT, IMPORT, and END assembly directives.
- [3] Set up a data area, then allocate and initialize variables in memory using data allocation directives.
- [3] Allocate and initialize constant data in code space.
- [2] Describe the alignment requirements of an ARM7TDMI microprocessor.
- [3] Use the load/store multiple instructions to perform block data moves.
- [3] Describe the function of the ADR/LDR pseudo-instructions and use them.
- [3] Write programs for the ARM7TDMI to perform specified functions using data movement, logical, and arithmetic instructions.

### Module 3

#### Assembly Language Programming, Stack

### Readings

6.12-6.13

- [2] Describe the operation of a look-up table and its uses.
- [3] Write look-up table code to perform a given transformation.
- [3] Use the logical instructions to perform bit-wise operations, including masking, testing, and toggling individual bits.
- [2] Describe the difference between arithmetic and logical shifts.
- [3] Use unconditional jumps to transfer program execution.
- [3] Write software to implement a jump table.
- [2] Describe the types of addressing modes that are available for branch instructions.
- [3] Use conditional execution and/or conditional branches to control program execution in order to perform a given task.
- [4] Determine the optimal use of conditional execution and/or branches to perform a given task.
- [3] Implement fixed-iteration and/or condition-based looping.
- [3] Use the ARM7TDMI instruction set to correctly implement the structured programming constructs of sequence, selection, and repetition.
- [4] Compare and contrast a hardware stack with a memory stack.
- [2] Describe the organization and operation of the ARM7TDMI stack.
- [3] Write code to initialize and use the stack.

#### Subroutines, Assembly Language Programming

- [3] Use the stack to save and restore registers and memory locations.
- [2] Describe the syntactical structure of a subroutine.
- [3] Write subroutines to perform given tasks.
- [4] Compare and contrast the different parameter-passing schemes.
- [3] Pass parameters using registers, memory, or the stack.
- [3] Write a subroutine using a standard stack frame to pass parameters and allocate local variables.
- [2] Describe the attributes of a reentrant subroutine in relation to parameter passing and local variable allocation.
- [3] Write code to perform multi-precision arithmetic operations.
- [3] Write code to convert a BCD value to binary.
- [3] Write code to convert an ASCII numeric string to binary.
- [3] Write code to convert binary data to an ASCII numeric string.

**Module 4**

## Microprocessor Support Circuits, I/O Subsystems

- [2] Describe the purpose and operation of the clock oscillator.
- [2] Describe how PLLs are used to generate clock signals.
- [4] Compare and contrast the clocking options available on the ADuC7026.
- [2] Describe how microprocessor operating modes are used to control power consumption.
- [2] Describe the purpose and operation of the reset circuit.
- [3] Describe the features commonly implemented in microprocessor supervisory circuits and their purpose/usage within the microprocessor system.
- [4] Compare and contrast time-multiplexed pins with configurable pins.
- [5] Use gate-level logic to design the basic structure of a GPIO pin.
- [5] Design simple input and output ports using MSI devices.
- [2] Describe the component parts of a generic IO device controller.
- [5] For a given scenario, design an appropriate IO device controller register model.
- [5] Design decoding logic for I/O devices.
- [4] Compare and contrast exhaustive, partial, and linear selection I/O decoding.
- [2] Describe the methods used to synchronize I/O transfers.
- [3] Implement conditional I/O using polling with timeout.
- [3] Configure and utilize the ADuC7026 GPIO pins.

## Microprocessor Peripheral Devices, Intro to ADuC7026

- [2] Identify the potential issues involved in interconnecting different logic families and/or using multiple voltage domains.
- [3] Calculate noise margins between different logic families and determine if they are compatible.
- [3] Determine the current that a driver must source/sink under different connection schemes.
- [3] Define and calculate fan-out.
- [2] Describe the causes and effects of capacitive loading.
- [3] For a given logic interconnection, determine if the logic is compatible.
- [2] Describe the purpose and operation of a periodic timer.
- [2] Describe the purpose and operation of a watchdog timer.
- [2] Describe the purpose and operation of a real-time clock.
- [3] Configure the ADuC7026 timers to meet given performance requirements.
- [2] Describe the purpose and operation of a PWM peripheral.
- [2] Describe the purpose and operation of a DMA controller.

**Readings**

7.1-7.4  
ADUC 51-  
52, 60-61,  
84-85  
Supp #2

ADUC 53-  
60, 71-73,  
75-79

**Module 5**

Memory System Design

**Readings**

7.5, 9  
ADUC 9-10,  
33-36, 43-  
47, 79-82

- [2] Describe the characteristics of ROM and RAM ICs.
- [2] Diagram the logical structure of a typical memory IC.
- [2] Describe the safeguards necessary to safely implement battery-backed SRAM.
- [2] Describe the circuitry, organization, and operation of typical SRAM, EPROM and flash memory devices.
- [3] Given a specific memory device, design a memory system of required width and depth.
- [5] Design a memory system to implement a given memory map for an 8-bit, 16-bit, 32-bit, or n-byte system.
- [5] Design a control signal scheme to support an n-byte memory system with byte-write capability, and describe/diagram its operation.
- [5] Design address decoding to implement a given memory map.
- [5] Determine the memory map implemented by a given address decoder design.
- [2] Describe the characteristic timing parameters for memory ICs during read and write cycles.

Memory Interfacing and Timing Analysis

Supp #3

- [2] Describe the operation of the ADuC7026 read and write bus cycles.
- [2] Describe the signals and timing associated with an ADuC7026 bus cycle.
- [4] Identify the features of the ADuC7026 control bus that support byte/word operations, and describe their operation.
- [2] Describe how the ADuC7026 address/data bus is de-multiplexed.
- [4] Describe how the width of the ADuC7026 external memory interface is controlled and how it impacts performance.
- [2] Describe the purpose and operation of wait states.
- [3] Configure the ADuC7026 external memory interface to meet given performance criteria.
- [4] Determine the timing requirements of the ADuC7026 external memory interface in different configurations.
- [4] For a given memory device or subsystem, analyze its timing characteristics and determine if the device or subsystem is compatible with an ADuC7026, and under what conditions.

## Module 6

### Interrupts and Exceptions

- [2] Describe the operation of a microprocessor in response to an interrupt.
- [2] Define in the context of interrupts: edge-sensitive, level-sensitive, priority, maskable, non-maskable, polled interrupts, vectored interrupts.
- [4] Compare and contrast polled and vectored interrupts.
- [2] Describe the operation of an interrupt vector table.
- [2] Describe the ARM7TDMI interrupt and exception modes.
- [2] Describe the requirements and considerations for writing ISRs.
- [3] Write ISRs to perform specified tasks.
- [2] Describe the typical uses of a non-maskable interrupt (NMI).
- [3] Configure and use the ADuC7026 interrupt system.
- [3] Implement semaphores to control simultaneous access to a shared resource.
- [2] Describe the use of software interrupts and exceptions.
- [2] Define interrupt latency.

### Readings

8  
ADUC 74-75  
ARM7 2.8-2.10

### Switches, Keypads, and Displays

- [5] Design circuitry to convert switch positions to logic levels.
- [2] Describe switch bounce, and the methods used to debounce switches.
- [3] Write a timer ISR to debounce a switch.
- [2] Describe the advantages of using matrix-connected switches.
- [3] Write software to scan a matrix-connected keypad.
- [2] Define roll-over, phantom key.
- [2] Describe the operation of rotary encoders, and the outputs available from quadrature, indexed, and absolute encoders.
- [2] Describe the electrical characteristics of an LED.
- [3] Draw the schematic diagram of a multiplexed multi-digit LED display, and describe its operation.
- [2] Define refresh rate and duty cycle for a multiplexed display.
- [3] Write an interrupt service routine to service a multiplexed display.
- [2] Describe the characteristics of LCD displays.

7.6-7.10  
Supp #5

**Module 7**

Serial Communications

- [4] Compare and contrast asynchronous and synchronous serial communications.
- [3] Draw and/or interpret the waveform of an asynchronous data frame, including start, data, parity, and stop bits.
- [2] Describe how center sampling (or multiple sampling) techniques are used to minimize errors.
- [2] Describe the basic structure and operation of a UART.
- [3] Program the ADuC7026 UART for a given serial frame format and baud rate.
- [3] Implement a circular queue in software.
- [2] Describe the signal levels and operation of the RS-232 interface.
- [2] Describe the operation of typical software and hardware flow control methods.
- [3] Write software to implement XON/XOFF flow control.

**Readings**

10  
ADUC 62-71  
Supp #4

Analog signals

- [2] Describe the components in a basic analog data acquisition system.
- [2] Define quantization, Nyquist frequency, and aliasing.
- [3] For an ADC, define resolution, accuracy, and aperture time.
- [2] Describe the relationship between ADC resolution and SNR.
- [2] Describe the purpose and implementation of anti-aliasing filters.
- [2] Describe the characteristics and operation of parallel (flash), successive approximation, dual-slope, and pipelined (multistage) ADCs.
- [2] Describe the characteristics of DACs.
- [2] For a DAC, define resolution, settling time, and glitch.
- [3] For a given DAC transfer characteristic, determine the errors (offset, gain, DNL, INL, monotonicity) present.
- [3] Implement a DAC using a filtered PWM output.
- [3] Describe the operation of an R-2R ladder DAC, and determine its output for a given bit combination.
- [2] Describe the purpose and implementation of reconstruction filters.
- [3] Implement sinusoidal waveform generation using phase accumulation, LUTs, and complex vector rotation.

11  
ADUC 30,  
37-42, 48-  
49