CmpE213 – Digital Systems Design

Learning Objectives
Wimp51 and CH1-2 MacKenzie

At the end of the following sections, you should be able to:

Wimp51:

1. Show movement of data through the processor for each instruction, labeling the fetch, decode, and execute phase. Predict the instruction executed if shown data movement.

2. Given a timing diagram of the clock, address, and data lines, identify a) the fetch, decode, and execute cycles b) the location in memory of code/data being loaded/written c) the ASM-code/value of code/data being loaded/written. Given an instruction to be loaded, create the timing diagram.

3. Define the primary operation performed by each instruction (e.g. Primary operation of "Addc A, #data" is : (A) < − (A) + #data + C)

4. Create/debug/analyze small ASM programs which accomplish specific tasks, given a listing of acceptable Wimp51 instructions (debug includes analysis and correction of mistakes). Show values in registers as the program executes. Predict total execution time.

5. Given the format of Wimp51 machine codes, convert an ASM program to a series of opcodes and vice-versa. Debug/analyze execution of a program given only the contents of code and data memory (ASM instructions not given, just the opcodes in code memory).

6. Draw an approximate timing diagram of "datapath" control signals inside the Wimp51, necessary to complete each instruction. Predict which instruction was executed given a timing diagram of control signals.

7. Make simple modifications to the Wimp51 architecture to perform simple new operations. Create opcodes for new operations that do not conflict with existing opcodes.

8. Explain the operation of a register, tri-state buffer, multiplexor, ALU, and CU (The purpose of each device, how control signals effect the device). Analyze a circuit using these devices (Show dataflow, inputs/outputs, control signals needed to produce events, etc). Explain the control/latching mechanism used by the Wimp51 registers. Apply understanding of operation of these devices in analysis of Wimp51 instruction datapaths.

9. List and define addressing modes used by the Wimp51. Given an Wimp51 instruction, identify the addressing mode(s) used.

Chapters 1, 2, + review: 8051 + hardware

1. List at least 5 real-world application of microcontrollers.
2. List at least 5 characteristics that change between members of the 8051 family. Explain why the 8051 is called a "family" of microcontrollers.
3. List the basic characteristics of Registers, RAM, and ROM. List the types of ROM and the basic characteristics of each.
4. Identify the name and purpose of the external pins on the 8051, given their "label" (e.g. P0, ALE, etc).
5. Describe the quasi-bidirectional nature of the I/O pins. Given the circuit diagram for an I/O pin, explain how data is read or written (how each component reacts to produce the result). With or without the diagram, explain why a 1 must be written to an I/O port for it to be used as an input. Given a sketch of transistors/resistors connected in parallel with an input port, identify whether the input port will read a 1 or 0.
6. Given a simple circuit with an 8051, a latch, and some external devices, identify the address space (range and type: code/data) of each device and any space which might be "wasted" (unavailable). Design an address decoding scheme to place devices in external code/data space and sketch the circuit connections. Addressing schemes may use: direct connect, simple logic gates, address decoders (74LS138), and/or PLDs. Explain when a chip is "selected" and how to read or write that chip (may involve CE, OE, RD, WR, PSEN, among others). Chips include: RAM, ROM, 82C55, and (to an extent) the 74LS138 and 74LS373. Given the circuit and the state of CE, OE, etc, identify which chip is selected and whether it is read/written. Use the exclusive nature of PSEN and (RD or WR) to analyze or design a circuit with overlapping code and data space addresses.
7. Explain the difference between internal memory and special function registers. Be able to show the differences (or lack thereof) between performing a direct-addr instruction like "MOV A, 42H" and the sequence "MOV R0,#42H","MOV A,@R0" (for values besides 42H) by a) showing the resulting contents of memory or b) given the contents of memory, showing the resulting content of the ACC. Explain why upper memory in the 8051 is split into two halves.
8. Recognize the difference between bit and byte memory. Identify the contents of byte memory after performing a bit instruction (e.g. say we perform the instruction SETB 42H. What byte memory location is effected? How?). Identify which SFRs are bit-addressable and at what addresses those bits are accessed.
9. Given a memory-map showing register banks and the contents of PSW, show which specific memory locations are accessed when executing register instructions – if reading a register, identify the value read; if writing, show the value written at its appropriate place in memory. Write code using the PSW to switch between register banks.
10. Identify the name and purpose of SFRs: A, B, SP, DPTR (DPH, DPL), P0-P3, PC (PCH, PCL), and PSW.

11. Given a list of instructions effecting flag settings and the format of PSW, show changes that occur to PSW when executing those instructions. Calculate the result of an arithmetic operation when the carry flag is set/cleared. Give the full names of the bits of PSW and identify their purpose, given only their labels (e.g. CY, P, AC, FO, etc).

12. Describe, using words and a circuit block-diagram with arrows showing signal-flow, the operations that are involved in reading/writing external devices (both in code and data space). Explain the advantage of multiplexing address and data lines. Explain the need for an external latch.

13. Given a timing diagram of RD, WR, ALE, PSEN, P0, and P2, identify a) any code or data which is read/written to/from memory b) how many machine cycles are represented c) any "dummy fetches". If code is read, given a list of instruction opcodes identify the specific instructions (eg. MOV A, #42 ; etc) loaded. Given an incomplete timing diagram of a memory read/write and the contents of memory, fill in the missing control/data signals. Explain *how* you know when a valid address or datum is available from the timing diagram (e.g. How do we know we have a valid address on P0, P2 when ALE goes from high to low? How do the control signals show if we are reading/writing data – explain). Be able to differentiate between a code-read, data-read, or data-write based on the timing diagram.