ECE199JL: Introduction to Computer Engineering **Fall 2012**
Notes Set 4

Summary of Part 4 of the Course

Most of the material in this part of the course is drawn from Patt and Patel Chapters 5 through 7.

In this short summary, we give you lists at several levels of difficulty of what we expect you to be able to do as a result of the last few weeks of studying (reading, listening, doing homework, discussing your understanding with your classmates, and so forth).

We'll start with the easy stuff. You should recognize all of these terms and be able to explain what they mean. For the specific circuits, you should be able to draw them and explain how they work.

- von Neumann elements
    - program counter (PC)
    - instruction register (IR)
    - memory address register (MAR)
    - memory data register (MDR)
    - processor data path
    - registers
    - bus
    - control signals
- instruction processing
    - fetch
    - decode
    - execute
    - register transfer language (RTL)
- Instruction Set Architecture (ISA)
    - encoding
    - fields
    - operation code (opcode)
    - types of instructions
        - operations
        - data movement
        - control flow
    - addressing modes
        - immediate
        - register
        - PC-relative
        - indirect
        - base + offset
- systematic decomposition
    - sequential
    - conditional
    - iterative
    - implementation in assembly
- assemblers and assembly code
    - opcode mnemonics (such as ADD, JMP)
    - two-pass process
    - symbol table
    - pseudo-ops
    - directives
- logic design optimization
    - bit-sliced (including multiple bits per slice)
    - serialized
    - pipelined logic
    - tree-based
- control unit design strategies
    - FSM implementation
    - hardwired control
        - single-cycle
        - multi-cycle
    - microprogrammed control
    - microinstruction
    - pipelining (of instruction processing)
    - use of MUXes and decoders

We expect you to be able to exercise the following skills:
- Implement arbitrary Boolean logic, and be able to reason about alternative designs in terms of their critical path delays and number of gates required to implement.
- Map RTL (register transfer language) operations into control signals for a given processor datapath.
- Systematically decompose a (simple enough) problem to the level of LC-3 instructions.
- Encode LC-3 instructions into machine code.
- Read and understand programs written in LC-3 assembly/machine code.
- Test and debug a small program in LC-3 assembly/machine code.

We expect that you will understand the concepts and ideas to the extent that you can do the following:

- Explain the basic organization of a computer's microarchitecture as well as the role played by elements of a von Neumann design in the processing of instructions.
- Identify the stages of processing an instruction (such as fetch, decode, getting operands, execution, and writing back results) in a processor control unit state machine diagram.
- Explain the role of different types of instructions in allowing a programmer to express a computation.
- Explain the tradeoffs in different addressing modes so as to motivate inclusion of multiple addressing modes in an ISA.
- Explain the importance of the three types of subdivisions in systematic decomposition (sequential, conditional, and iterative).
- Explain the process of transforming assembly code into machine code (that is, explain how an assembler works, including describing the use of the symbol table).

At the highest level, we hope that, while you do not have direct substantial experience in this regard from our class, that you will nonetheless be able to begin to do the following when designing combinational logic:

- Design and compare implementations using gates, decoders, muxes, and/or memories as appropriate, and including reasoning about the relevant design tradeoffs in terms of area and delay.
- Design and compare implementation as a bit-sliced, serial, pipelined, or tree-based design, again including reasoning about the relevant design tradeoffs in terms of area and delay.
- Design and compare implementations of processor control units using both hardwired and microprogrammed strategies, and again including reasoning about the relevant design tradeoffs in terms of area and delay.