

ECE120: Introduction to Computer Engineering

Notes Set 4.4 Summary of Part 4 of the Course

With the exception of control unit design strategies and redundancy and coding, most of the material in this part of the course is drawn from Patt and Patel Chapters 4 through 7. You may also want to read Patt and Patel's Appendix C for details of their control unit design.

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.

- von Neumann elements
 - program counter (PC)
 - instruction register (IR)
 - memory address register (MAR)
 - memory data register (MDR)
 - processor datapath
 - bus
 - control signal
 - instruction processing
- Instruction Set Architecture (ISA)
 - instruction encoding
 - field (in an encoded instruction)
 - operation code (opcode)
- assemblers and assembly code
 - opcode mnemonic
(such as ADD, JMP)
 - two-pass process
 - label
 - symbol table
 - pseudo-op / directive
- systematic decomposition
 - sequential
 - conditional
 - iterative
- control unit design strategies
 - control word / microinstruction
 - sequencing / microsequencing
 - hardwired control
 - single-cycle
 - multi-cycle
 - micropipelined control
- error detection and correction
 - code/sparse representation
 - code word
 - bit error
 - odd/even parity bit
 - Hamming distance between code words
 - Hamming distance of a code
 - Hamming code
 - SEC-DED

We expect you to be able to exercise the following skills:

- Map RTL (register transfer language) operations into control words 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.
- Be able to calculate the Hamming distance of a code/representation.
- Know the relationships between Hamming distance and the abilities to detect and to correct bit errors.

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

- Explain the role of different types of instructions in allowing a programmer to express a computation.
- 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).
- Be able to use parity for error detection, and Hamming codes for error correction.

At the highest level, we hope that, while you do not have direct substantial experience in this regard from our class (and should not expect to be tested on these skills), 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.
- Understand basic tradeoffs in the sparsity of code words with error detection and correction capabilities.