Systems Architecture

Lecture 2: Implementation of a Simplified Computer

Jeremy R. Johnson
William M. Mongan
Introduction

- Objective: To develop a simple model of a computer and its execution that is capable of executing RAM programs. To introduce the concept of abstraction in computer design.

- The model will be given schematically with timing sequences.
- RAL instructions will be implemented using microinstructions described in a notation called “Register Transfer Language” (RTL).
- The control logic for implementing microinstructions will be described at the gate level.

SCRAM

- A Simple but Complete Random Access Machine. This computer can execute RAL instructions.
- 8-bit words
- 16 word memory (4 address bits)
- Instructions (4 bit opcode, 4 bit operand)
- 7 registers
  - PC (program counter)
  - IR (instruction register - IR(C) = instruction code, IR(O) = operand
  - MAR (memory address register)
  - MBR (memory buffer register)
  - AC (accumulator)
  - AD (register for addition internal to the ALU - arithmetic logic unit)
- Driven by the CLU (control logic unit)
- A timer T generates pulses that are decoded into separate input lines to the CLU
Fetch and Execute

• A cycle of operation consists of two stages
  – The fetch cycle gets the next executable instruction and loads it into the IR
  – The execute cycle performs the instruction in the IR

• The fetch and execute cycles are written as a sequence of micro-instructions described in a notation called “Register Transfer Language” (RTL)

• Important: this machine uses a timer T that “ticks” several times per each of the two cycles; therefore, the fetch and execute cycles consist of several clock cycles.
Instruction Opcodes

- **LDA** 0001 \( X \); Load contents of memory address \( X \) into the AC
- **LDI** 0010 \( X \); Indirectly load contents of address \( X \) into the AC
- **STA** 0011 \( X \); Store contents of AC at memory address \( X \)
- **STI** 0100 \( X \); Indirectly store contents of AC at address \( X \)
- **ADD** 0101 \( X \); Add contents of address \( X \) to the AC
- **SUB** 0110 \( X \); Subtract contents of address \( X \) from the AC
- **JMP** 0111 \( X \); Jump to the instruction labeled \( X \)
- **JMZ** 1000 \( X \); Jump to instruction \( X \) if the AC contains 0
Microprogram

• Fetch cycle
  - $t_0$: MAR $\leftarrow$ PC
  - $t_1$: MBR $\leftarrow$ M; PC $\leftarrow$ PC + 1
  - $t_2$: IR $\leftarrow$ MBR

• Execute cycle (LDA)
  - $q_1t_3$: MAR $\leftarrow$ IR(O)
  - $q_1t_4$: MBR $\leftarrow$ M
  - $q_1t_5$: AC $\leftarrow$ MBR
Microprogram

• Execute cycle (LDI)
  - $q_2t_3$: MAR $\leftarrow$ IR(O)
  - $q_2t_4$: MBR $\leftarrow$ M
  - $q_2t_5$: MAR $\leftarrow$ MBR
  - $q_2t_6$: MBR $\leftarrow$ M
  - $q_2t_7$: AC $\leftarrow$ MBR

• Execute cycle (ADD)
  - $q_5t_3$: MAR $\leftarrow$ IR(O)
  - $q_5t_4$: MBR $\leftarrow$ M
  - $q_5t_5$: AD $\leftarrow$ MBR
  - $q_5t_6$: AD $\leftarrow$ AD + AC
  - $q_5t_7$: AC $\leftarrow$ AD
Control Logic for the Fetch Cycle

- $t_0$: MAR $\leftarrow$ PC
- $t_1$: MBR $\leftarrow$ M; PC $\leftarrow$ PC + 1
- $t_2$: IR $\leftarrow$ MBR
Logic for Loading the Accumulator

\[ q_1 \]

\[ t_3 \] \rightarrow \text{Logic gate} \rightarrow \text{Logic gate} \rightarrow X_{10} \quad \text{MAR} \leftarrow \text{IR(0)}

\[ t_4 \] \rightarrow \text{Logic gate} \rightarrow \text{Logic gate} \rightarrow X_4 \quad \text{X_4}

\[ t_5 \] \rightarrow \text{Logic gate} \rightarrow \text{Logic gate} \rightarrow X_5 \quad \text{MBR} \leftarrow \text{M}

\[ X_{11} \quad \text{AC} \leftarrow \text{MBR} \]

\[ X_{12} \]
CLU Logic

- Some of the output lines from the two previous slides appear in both circuits. It is necessary to have some logic to connect and coordinate the individual outputs to the wires leaving the CLU.
Exercises

- Write microprograms for STA, STI, and JMZ. Implement the microprograms in standard logic.

- Design the portion of the CLU that determines the two output lines labeled $x_{10}$. Input to this circuit will be one or both of the lines previously labeled $x_{10}$ in the individual circuits for LDA, LDI, and the other circuits.

- Convert the following program to the equivalent set of binary words, as indicated in this chapter. This is called machine code. Trace the execution of the program by listing the q, t, and x variables.
  - LDA 1
  - ADD 2
  - STA 3