LC-3
Instruction Processing

(Textbook’s Chapter 4)

Next set of Slides:
Textbook Chapter 10-10.2
Instruction Processing

• It is impossible to do all of an instruction in one clock cycle.
• Processors break an instruction into standard steps called phases.
Overview

• Phases of instruction execution
• Major components of LC-3 processor
• Consider several instructions and how the LC-3 executes instructions using those components
Instruction Processing

1. Fetch instruction from memory
2. Decode instruction
3. Evaluate address
4. Fetch operands from memory
5. Execute operation
6. Store result
Phases of instruction processing

Six basic *phases* of instruction processing:

\[ F \rightarrow D \rightarrow EA \rightarrow OP \rightarrow EX \rightarrow S \]

- **NOTE:**
  - Not all phases are needed by every instruction
  - All instructions will go through F and D at least
  - Phases may take more than 1 clock cycle (if they involve memory)
FETCH

Load next instruction (at address stored in \( \text{PC} \)) from memory into Instruction Register (IR).

- Copy contents of \( \text{PC} \) into MAR.
- Send “read” signal to memory.
- Copy contents of MDR into IR.

Then increment \( \text{PC} \), so that it points to the next instruction in sequence.

- \( \text{PC} \) becomes \( \text{PC} + 1 \).
DECODE

First identify the opcode.

– In LC-3, this is always the first four bits of instruction.
– A 4-to-16 decoder asserts a control line corresponding to the desired opcode.

Depending on opcode, identify other operands from the remaining bits.

– Example:
  • for `LDR`, last six bits is offset
  • for `ADD`, last three bits is second source operand
EVALUATE ADDRESS

For instructions that require memory access, compute address used for access.

Examples:
- add offset to base register (as in LDR)
- add offset to PC
- add offset to zero
- set source registers addresses
FETCH
OPERANDS

Obtain source operands needed to perform operation.

Examples:
- load data from memory (LDR)
- read data from register file (ADD)
EXECUTE

Perform the operation, using the source operands.

Examples:
- send operands to ALU and assert ADD signal
- do nothing (e.g., for loads and stores)
STORE RESULT

Write results to destination. (register or memory)

Examples:
- result of **ADD** is placed in destination register
- result of memory load is placed in destination register
- for store instruction, data is stored to memory
  - write address to MAR, data to MDR
  - assert WRITE signal to memory
LC-3 Data Path

Filled arrow = info to be processed.

Unfilled arrow = control signal.
Data Path Components (1)

Global bus
- special set of wires that carry a 16-bit signal to many components
- inputs to the bus are “tri-state devices,” that only place a signal on the bus when they are enabled
- only one device “speaks” on the bus at any given time
  - control unit decides which signal “drives” the bus
- any number of components can read the bus
  - the control unit write-enables the destination device
LC-3
Global Bus

16 bits wide

Output drivers control source

Input signals control destination

Not all I/O signals to global bus are shown
Data Path Components (1A)

Memory

– Control and data registers for memory and I/O devices
– memory: MAR, MDR (also control signal for read/write)
LC-3 Memory and I/O systems

Memory Components:
- Memory
- Mem. Address Reg.
- Mem. Data Reg.
- Mem Enable, R, W

Input and Output
(has same signals as memory but not shown)
Data Path Components (2)

ALU

- Accepts inputs from register file and from sign-extended bits from IR (immediate field).
- Output goes to bus, and is used by condition code logic, register file, memory

Register File

- Two read addresses (SR1, SR2), one write address (DR)
- Input from bus
  - result of ALU operation or memory read
- Two 16-bit outputs
  - used by ALU, PC, memory address
  - data for store instructions passes through ALU
LC-3
Register files and ALU

Register File
Registers R0-R7
16 bits Data in
2 x 16 bits Data out
10 control bits

ALU and SR2MUX
Data: 3 16-bit words
Control: 2 bits for ALU and 1 bit for MUX
Data Path Components (3)

PC and PCMUX
- There are three inputs to PC, controlled by PCMUX
  1. PC+1 – FETCH stage
  2. Address adder – **BR, JMP**
  3. bus – **TRAP** (discussed later)

MAR and MARMUX
- There are two inputs to MAR, controlled by MARMUX
  1. Address adder – **LD/ST, LDR/STR**
  2. Zero-extended IR[7:0] -- **TRAP** (discussed later)
LC-3
PC and MARMUX

PC; PCMUX; incremengt
PC gets data from 1 of 8 sources.

MAR and MARMUX
MAR gets data from how many sources?
Data Path Components (4)

Condition Code Logic
- Looks at value on bus and generates N, Z, P signals
- Registers set only when control unit enables them (LD.CC)
  - only certain instructions set the condition codes
    (ADD, AND, NOT, LD, LDI, LDR, LEA)

Control Unit – Finite State Machine
- On each machine cycle, changes control signals for next phase of instruction processing
  - who drives the bus? (GatePC, GateALU, …)
  - which registers are write enabled? (LD.IR, LD.REG, …)
  - which operation should ALU perform? (ALUK)
  - …
- Logic includes decoder for opcode, etc.
LC-3
C.Code
Registers and FSM

Condition Code Logic
Sets codes when LD.CC allows.

Control Unit
Is an FSM
Controls everything based on instruction and CC.
Data Path Components (5)

Effective Address Mux and Adder

- Computes addresses for PC and MAR
- Used for most instructions that affect the PC or access memory
  (BRA, JMP, LD, ST, JSR, etc.)
Effective Address Computation Units: For almost all PC and MAR computations.
Example 1:

```
x30A2 add R2, R0, R1
```

```
R0 = x0230
R1 = x1111
R2 = x2222
```

1) Fetch (from mem) 1st step
Example 1:

x30A2 add R2, R0, R1

R0 = x0230
R1 = x1111
R2 = x2222

1) Fetch
2nd step
Example 1:

\[ x_{30A2} \text{ add } R2, R0, R1 \]

\[
\begin{align*}
R0 &= x_{0230} \\
R1 &= x_{1111} \\
R2 &= x_{2222}
\end{align*}
\]

2) Decode
Example 1:

```
x30A2 add R2, R0, R1
```

R0 = x0230
R1 = x1111
R2 = x2222

5) Execute
Example 1:

\[ \text{x30A2 add R2, R0, R1} \]

\[ \begin{align*}
R0 &= x0230 \\
R1 &= x1111 \\
R2 &= x1341
\end{align*} \]

6) Store Results

\[ \begin{align*}
x0230 &\quad x1111 &\quad x1341
\end{align*} \]
Example 2:

\[ x_{3117} \text{ LDR R3, R5, } x_B \]

\[ \begin{align*}
R3 &= x_{3451} \\
R5 &= x_{8800}
\end{align*} \]

1) Fetch 1st step
Example 2:

X3117 LDR R3, R5, xB

R3 = x3451
R5 = x8800

1) Fetch
2nd step
Example 2:

```
X3117 LDR R3, R5, xB
```

\[
R3 = x3451 \\
R5 = x8800
\]

2) Decode

```
x674B
```

\[
x674B
\]
Example 2:

4) Evaluate Address

X3117 LDR R3, R5, xB

R3 = x3451
R5 = x8800

x880B

x000B

x0B
Example 2:

X3117 LDR R3, R5, xB

R3 = x3451
R5 = x8800

4) Fetch Operands

X????
Example 2:

X3117 LDR R3, R5, xB

R3 = \text{mem}[x880B]
R5 = x8800

6) Store Results
Example 3:

\[ x_{3040} \text{ BRZ EndLoop} \]

EndLoop = \( x_{3030} \)

Therefore

\[ \text{mem}[x_{3040}] = x_{05EF} \]

1) Fetch

1\textsuperscript{st} step
Example 3:

- **x3040 BRZ EndLoop**
- **EndLoop = x3030**

1) Fetch
2nd step

- **x3041**
- **x05EF**
Example 3:

x3040 BRZ EndLoop

2) Decode
Example 3:

```
x3040 BRZ EndLoop
```

```
x3030
```

```
xFFEF (16bits)
x1EF (9bits)
```

```
x3041
```

3) Evaluate Address
Example 3:

x3040 BRZ EndLoop

xFFEF (16bits)

(no Operand Fetch)

(no Execute)

6) Store Results
Example 4:

X3317 STR R3, R5, xB

R3 = x3451
R5 = x8800

1) Fetch 1st step
Example 4:

\[ X3117 \text{ STR R3, R5, xB} \]

\[ R3 = x3451 \]
\[ R5 = x8800 \]

1) Fetch
2nd step
Example 4:

**x3117 STR R3, R5, xB**

R3 = x3451  
R5 = x8800

2) Decode
Example 4:

\[ \text{x3117 STR R3, R5, xB} \]

\[ R3 = \text{x3451} \]
\[ R5 = \text{x8800} \]

4) Evaluate Address

\[ \text{x000B} \]
\[ \text{x001011} \]

\[ \text{x774B} \]
\[ \text{x880B} \]
Example 4:

```
X3117 STR R3, R5, xB
```

R3 = x3451
R5 = x8800

ALU told to AND; AND(x3451, x3451) = x3451

4) Fetch Operands
Example 4:

\[ \text{x3117 STR R3, R5, xB} \]

\[ \text{R3 = x3451} \]
\[ \text{R5 = x8800} \]
\[ \text{mem[x880B] <- x3451} \]

No Execute phase

6) Store Results

\[ \text{x3451} \]
Recommended and **Homework exercises for Nov. 16:**

- **Recommended:** Ex 4.8, 4.10
- **Recommended:** Ex 4.13 and 4.16 (a little bit more advanced)
- **To Turn in:** For the instruction LD R1, NAME, show the Evaluate Address phase of the instruction by showing the inputs and outputs of the Effective Address Computation Units. Instruction at x3012 NAME at x3001