A Simple Processor Architecture

Decoder

Register 1
Register 2
Register 7

Destination Select

B Select
Multiplexer

B Bus

ALU

Function Select

Shift Select

Output

Input

A Select

A Bus

Multiplexer

Martin B.H. Weiss

Arithmetic-Logic Unit and Processor Design - 1
University of Pittsburgh
ALU

- Performs Arithmetic Functions
- Performs Logic Functions
- Function is Selected by Control
- Status Bits
  - C - Carry
  - V - Overflow
  - Z = 1 If Resultant Contains All Zeros
  - S - Sign Bit of the Result
- Decoder Selects Destination for the Resultant
**ALU**

- **Inputs**
  - Operands
  - Input Carry
  - Operation Select
    - Add
    - Subtract
    - AND
    - OR
    - XOR
  - Mode (Arithmetic or Logic) Select

- **Outputs**
  - Resultant
  - Output Carry
## ALU Function Table

<table>
<thead>
<tr>
<th>Operation Select</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_2$ $S_1$ $S_0$ $C_{in}$</td>
<td>$F=A$</td>
<td>Transfer A</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>$F=A+1$</td>
<td>Increment A</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>$F=A+B$</td>
<td>Add A and B</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>$F=A+B+1$</td>
<td>Add A and B With Carry</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>$F=A+B'$</td>
<td>Add A and One’s Complement of B</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>$F=A-1$</td>
<td>Decrement A</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>$F=A+B'+1$</td>
<td>Subtract B From A</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>$F=A-1$</td>
<td>Decrement A</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>$F=A$</td>
<td>Transfer A</td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>$F=AB$</td>
<td>AND</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>$F=A+B$</td>
<td>OR</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>$F=A \text{ XOR} B$</td>
<td>Exclusive OR</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>$F=A'$</td>
<td>Complement</td>
</tr>
</tbody>
</table>
ALU Components

- Arithmetic
  - Parallel Add
  - One Full Adder per Bit
  - Selection Logic
- Logic
  - Gates
  - Multiplexer
Circuit for Arithmetic Component

C_{in}
S_0
S_1
A_1
B_1

A_2
B_2

X_1
C_1

Full Adder
F_1

Y_1

X_2
C_2

Full Adder
F_2

Y_2

Martin B.H. Weiss
University of Pittsburgh
Circuit for Logic Component

MUX

$F_n$

$S_0$

$S_1$

$A_n$

$B_n$
Internal Structure of ALU

Arithmetic Stage

Logic Stage

Multiplexer

$C_n$-1

$A_n$

$B_n$

$S_0$

$S_1$

$S_2$

$F_n$
Shifter

- General
  - Extension of Shift Register Circuit is Possible
  - This Requires Several Clock Pulses
  - This is Time Consuming
- Alternate Approach (Figure 7-18, p. 246 of Mano)
  - Use Multiplexers
  - Wire to Cause Shift Effect
  - Control Determines Nature of Shift
  - Thus, a Single Clock Cycle is Used
Control Unit Requirements

- MUX A Selector
- MUX B Selector
- ALU Operation Selector
- Shift Selector
- Destination Selector
Control Word

- Number and Organization of Bits Required to Control ALU
- Bit Requirements
  - A: A Bus Select (Seven Registers Plus Input): 3 bits
  - B: B Bus Select (Seven Registers Plus Input): 3 bits
  - D: Destination Select (Seven Registers): 3 bits
  - F: ALU Control (Four bits)
  - H: Shift Control (Three bits)
  - TOTAL = 16 bits
  - Thus, 16 Bits Can Be Used to Perform All Microoperations
# Control Word Encoding

<table>
<thead>
<tr>
<th>Code</th>
<th>Operation (F)</th>
<th>$C_{in} = 0$</th>
<th>$C_{in} = 1$</th>
<th>A</th>
<th>B</th>
<th>D</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>$F = A$</td>
<td>$F = A + 1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Input</td>
</tr>
<tr>
<td>001</td>
<td>$F = A + B$</td>
<td>$F = A + B + 1$</td>
<td></td>
<td>R1</td>
<td>R1</td>
<td>R1</td>
<td>SHL</td>
</tr>
<tr>
<td>010</td>
<td>$F = A + B'$</td>
<td>$F = A + B' + 1$</td>
<td></td>
<td>R2</td>
<td>R2</td>
<td>R2</td>
<td>SHR</td>
</tr>
<tr>
<td>011</td>
<td>$F = A - 1$</td>
<td>$F = A$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Bus=0</td>
</tr>
<tr>
<td>100</td>
<td>$F = AB$</td>
<td></td>
<td></td>
<td>R4</td>
<td>R4</td>
<td>R4</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>$F = A + B$</td>
<td></td>
<td></td>
<td>R5</td>
<td>R5</td>
<td>R5</td>
<td>ROL</td>
</tr>
<tr>
<td>110</td>
<td>$F = A \ XOR \ B$</td>
<td></td>
<td></td>
<td>R6</td>
<td>R6</td>
<td>R6</td>
<td>ROR</td>
</tr>
<tr>
<td>111</td>
<td>$F = A'$</td>
<td></td>
<td></td>
<td>R7</td>
<td>R7</td>
<td>R7</td>
<td></td>
</tr>
</tbody>
</table>
Microoperations and Microprograms

- Example Microoperation
  - $R1 \leftarrow R2 - R3$
  - Symbolically: $R2, R3, R1, F = A - B, \text{No Shift}$
  - Control Word = 010 011 001 0101 000 = 4CA8 (H)

- Clearly, Many Microoperations Are Possible

- Control Memory
  - Location of Available Microoperations
  - Width of Control Memory = Control Word

- Microprograms Can be Written Using a Sequence of Microoperations
Addressing Modes

- **Direct**
  - the address is contained in the address field
  - the size of the memory is limited by the size of the address field

- **Indirect**
  - Content of the Memory Location Contained in the Address Field Points to The Actual Address
  - Allows for a Larger Memory Because the Address Fields Can Be Larger
  - Allows for Efficient Address Manipulation

- **Indexed**
  - Content of the Memory Field is Added to the Contents of the Index Register
  - Allows for Flexible Relative Addressing
Control System Design

- A Programmer Needs a Logical Structure and Instructions
- A Hardware Designer has Microoperations
- The Bridge Between These is a Microprogram
Structure of a Microprogrammed Control Unit

Input

Next Address Generator

Control Address Register

Control Memory (ROM)

Control Data Register

Control Word
Example

- General
  - 16 Bits for Processor Control (Bits 1-16; A,B,D,F,H As Before)
  - One Bit for Address Source Selection (bit 17)
  - Three Bits For Status Bit Select (Bits 18-20)
  - Six Bits for the Next Address (Bits 21-26)
- \(2^6 = 64\) Microinstructions Are Possible
- CAR Can be Loaded or Incremented, Depending on the Condition
Example

External Address

Mux

Control Address Register

Control Memory

1-16
17
18-20
21-26

Increment
Load

Multiplexer

0 1 C C' Z Z' S V

Input Data

Processor Unit

Output Data

Control Word
A Simple Computer Design

Clock

CPU

Memory

I/O

Address Bus

Data Bus

Interrupt

Read/Write

Tele 2060

Martin B.H. Weiss

University of Pittsburgh
A Simple Computer Design

- I/O Interface
- 16 Bit Data Bus
- 16 Bit Address Bus
CPU Components

Desti-

nation

Decoder

Fourteen Registers
(16 Bits)

Mux A

Mux B

ALSU
(16 Bits)

Input

Output

V S Z C

Martin B.H. Weiss

Computer Organization - 16

University of Pittsburgh
CPU Components

- ALSU
- Registers
- Internal Busses
- Status Bits
ALSU

- Two 16 Bit Input Busses
- Four Units Multiplexed Together
- Each Unit Always Operates on Both Operands
- The Operation is Determined by the Selection Lines $S_0$ and $S_1$
- Operation is Selected by a 4:1 Multiplexer ($S_2$ and $S_3$)
- 16 Operations are Possible on the Output
- Example 0110 => F = A B
- Requires a Five Bit Control Word
## ALSU Function Table

<table>
<thead>
<tr>
<th>Operation Select</th>
<th>Function</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_{in} = 0$</td>
<td>$C_{in} = 1$</td>
</tr>
<tr>
<td>$S_3$ $S_2$ $S_1$ $S_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>$F=A$</td>
<td>$F=A+1$</td>
</tr>
<tr>
<td>0 0 0 1</td>
<td>$F=A+B$</td>
<td>$F=A+B+1$</td>
</tr>
<tr>
<td>0 0 1 0</td>
<td>$F=A-B -1$</td>
<td>$F=A-B$</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>$F=A-1$</td>
<td>$F=A$</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>$F=AB$</td>
<td></td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>$F=A+B$</td>
<td></td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>$F=A \text{ XOR } B$</td>
<td></td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>$F=A'$</td>
<td></td>
</tr>
<tr>
<td>1 0 0 0</td>
<td>$F=\text{shr } A$</td>
<td></td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>$F=\text{ror } A$</td>
<td></td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>$F=\text{ror } A \text{ w. Carry}$</td>
<td></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>$F=\text{asr } A$</td>
<td></td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>$F=\text{shl } A$</td>
<td></td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>$F=\text{rol } A$</td>
<td></td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>$F=\text{rol } A \text{ w. Carry}$</td>
<td></td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>$F=\text{asl } A$</td>
<td></td>
</tr>
</tbody>
</table>
Registers

- Fourteen Total Registers
- PC
- Six General Purpose Registers
- Seven Special Purpose Registers
  - Index Register
  - Stack Pointer
  - Source Register
  - Destination Register
  - Temporary Register
  - Two Constant Registers (Zero and N=16)
Three Internal Busses

- One for Each Operand
  - Attached to One of the Registers via a Multiplexer
  - One Mux for Each Operand Bus
- One For the Result
  - Destination is One of the Registers
  - May Also Be External
Control Word

- 17 Bits Long
- Five Bits for ALSU (Four Control + C_{in})
- Four Bits for Mux A
- Four Bits for Mux B
- Four Bits for Destination Decoder
Organization of the CPU

Data Bus

Bus Drivers

DIR

DOR

IR

Control Unit

Processor Unit

AR

Address Bus

CPU
CPU

- Components
  - Processor Unit
  - Memory
  - Control Unit
  - Buffers/Registers
  - Busses

- Busses
  - External to the Processor
  - Data Bus (16 bits)
    - Direction Must be Mediated
    - Read/Write Line (From/To Memory)
  - Address Bus (16 bits)
CPU Buffers and Registers

- **Data Input Register**
  - Input to IR or Processor
  - Data from Memory or From the Outside World
- **Data Output Register** - Output to Memory or Outside World
- **Address Register** - Current Memory Address
- **Instruction Register** - OPcode of the Current Instruction
Control Unit

- Components
  - Subroutine Register
  - OPCode Map
  - Incrementer
  - Address Mux
  - Control Address Register
  - Control Memory
- Subroutine Register
  - Used in Subroutine Call
  - Stores Return Address
Control Unit

- Opcode Map
  - Maps OPcode Into Control Memory Address
  - May be a ROM
- Incrementer - Increments Address
- Address Mux
  - Selects Source of Address
  - Controlled by Control Memory (CS field)
- Control Address Register - Holds the Memory Address
- Control Memory - Contains the Microcode
Instruction Types

Indirect Bits

<table>
<thead>
<tr>
<th>Type 0</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI</td>
<td>DI</td>
<td>SI</td>
<td>SI</td>
</tr>
<tr>
<td>0, 0</td>
<td>0, 1</td>
<td>1, 0</td>
<td>1, 1</td>
</tr>
<tr>
<td>SRC</td>
<td>SD</td>
<td>0, 0</td>
<td>0, 0</td>
</tr>
<tr>
<td>DST</td>
<td>MDI</td>
<td>0, 0</td>
<td>0, 0</td>
</tr>
</tbody>
</table>

- **Register to Register**
- **Memory to Register**
- **Branch**
- **Implied**
Type 1

- Memory-Register
- Two Word Instruction (Except for One Operand Instructions with Operand in Register)
- Format
  - Bits 0-2: Register
  - Bit 3: Indirect Bit
  - Bits 4-5: Addressing Mode
    - Immediate: W is the Operand
    - Direct
    - Indirect
    - Index
  - Bit 6-7: Source/Destination and Number of Operands
    - Memory or Register
    - One or Two Operands
Type 1

- Format
  - Bits 8-13: OPCode
  - Bits 14-15: 01 (Indicates Memory-Register Operation)
  - Second Word for Memory Address or Intermediate Operand
Type 2

- Branch
- Two Word Instruction
- Format
  - Bits 0-7: 0
  - Bits 8-13: OPCODE
  - Bits 14-15: 10 (Indicates Branch Operation)
- Second Word Contains the Branch Address
Type 3

- Implied Mode
- Operand Either Does Not Exist or is Implicit in the Instruction
- Example: NOP
- One Word
- Format
  - Bits 0-7: 0
  - Bits 8-13: OP code
  - Bits 14-15: 11
An Assembly Language Instruction: ADD

- Adds Two Numbers
- May be Type 0 or Type 1
  - Type 0 if the Operands are in Registers
  - Type 1 if One of Them is in Memory
- Example of Type 0: ADD R5,R2
  - Operation: R2 ← R2+R5
  - Machine Code (0952H)
    - Bits 0-2: 010 (Register 2)
    - Bit 3: Indirect Bit = 0 (Direct)
    - Bits 4-6: 101 (Register 5)
    - Bit 7: Indirect Bit = 0 (Direct)
    - Bits 8-13: OPCode = 001001
    - Bits 14-15: 00 (Indicates Register-Register Operation)
- Note That All Type 1 ADD Instructions Have 09H As First Byte
Alternate Form of ADD

- If We Used Indirect in R2: ADD R5,(R2)
  - Machine Code:
    - Bits 0-2: 010 (Register 2)
    - Bit 3: Indirect Bit = 1 (Direct)
    - Bits 4-6: 101 (Register 5)
    - Bit 7: Indirect Bit = 0 (Direct)
    - Bits 8-13: OPCODE = 001001
    - Bits 14-15: 00 (Indicates Register-Register Operation)
  - Alternatively: 095AH
Example of a Type 1 Addition: ADD TEMP,R2

- \( R2 \leftarrow R2 + M[\text{TEMP}] \)
- Machine Code
  - Bits 0-2: 010
  - Bit 3: Indirect bit = 0
  - Bits 4-5: Addressing Mode = 00
  - Bit 6-7: Source/Destination and Number of Operands = 00
  - Bits 8-13: OPCode = 001001
  - Bits 14-15: 01
- Machine Code: 4902H
Notes

- Many Other Instructions Exist as Well
- We Will Examine the Intel 8080 Later
- The Software That Translates Assembly Code to Machine is an Assembler
Microinstructions

- 23 bits
- 17 for ALSU
  - Operand A Select (4)
  - Operand B Select (4)
  - Destination Select (4)
  - ALSU Function Select (5)
- Six for Other Functions
  - Two for Control Sequence
    - Distinguishes Microinstruction Formats
    - Controls the Address Mux in the Control Unit
  - Four for Miscellaneous Microoperations
Example: $R2 \leftarrow R2 + R5$

- Refer to Mano, Figure 10-8(b) (p. 349)
- Control Sequence (CS) = 00
- Register Select “A” (AS) = 0010
- Register Select “B” (BS) = 0101
- Destination Select (DS) = 0010
- Function Control (FC) = 00010
- Miscellaneous (MS) = 0000
- Hex code: 04A620
- Microprograms Map OPcode Semantics into Microoperation Sequences
Microprogram Flowchart for Type 0 ADD

Source Indirect

\[ AR \leftarrow R(S) \]

\[ SR \leftarrow DIR \]

Source Direct

\[ SR \leftarrow R(S) \]

Destination Indirect

\[ AR \leftarrow R(D) \]

\[ DOR \leftarrow SR + DIR \]

\[ WRITE \text{ and Check for Interrupt} \]

\[ IR7 \]

\[ 1 \rightarrow 0 \]

Destination Direct

\[ IR3 \]

\[ 1 \rightarrow 0 \]

\[ R(D) \leftarrow SR + R(D) \]

Check for Interrupt

ADD (Type 0)
Program Status Word (PSW)

- General
  - Contains Important CPU Status Indications
  - Need Sufficient Information to Restore Processor Context (Status)
- Typical Contents
  - Program Counter
  - Stack Pointer
  - Accumulator Value
  - Index Register(s)
  - Processor Status Register
  - Instruction Register
Stack

- Special Type of Memory
- Operations
  - PUSH
  - POP
Serial Communications Devices

Control
Address Bus
Data Bus

Data Bus

Control

Transmit Shift Register
Transmit Holding Register
Receive Holding Register
Receive Shift Register
Computer Architecture - Intel 8080

- Defines Only the CPU
- Physical Features
  - 40 Pin DIP Package (See Handout)
  - 8 Bit Data Bus
  - 16 Bit Address Bus
  - Control Pins
Logical Features

- 8 Bit ALU
- Registers
  - ACC
  - PC
  - Stack Pointer
  - IR
  - 6 Working Registers
    - Used in Pairs
    - B-C
    - D-E
    - H-L
  - 2 Temporary Registers
- Organization (See Handout)
Instruction Format

- 8 Bit Opcode
- 256 Instructions
- 8080 Instructions (See Handout)
- 8080 Microinstructions (See Handout)
Summary of Digital and Computer Section

- Digital Computers Consist of Sequential and Combinational Circuits
- General Purpose Devices That Can Be Programmed
- Control is Often Implemented Via Microprograms
Modem Revisited

Async. to Sync. -> Encoder (Error) -> Scrambler -> Low Pass Filter -> Modulator -> Bandpass Filter

Control

Clock

Hybrid

Receiver

Sync. to Async. -> Decoder (Error) -> Descrambler -> Low Pass Filter -> Demodulator -> Bandpass Filter

Transmitter