

Hardwired Control Unit

Ch 14

Micro-operations
Controlling Execution
Hardwired Control

11/10/2000 Copyright Teemu Kerola 2000 1

What is Control ⁽²⁾

- So far, we have shown what happens inside CPU
 - execution of instructions
 - opcodes, addressing modes, registers
 - I/O & memory interface, interrupts
- Now, we show how CPU controls these things that happen
 - how to control what gate or circuit should do at any given time
 - control wires transmit control signals
 - control unit decides values for those signals

11/10/2000 Copyright Teemu Kerola 2000 2

Micro-operations ⁽²⁾ (micro-operation)

- Basic operations on which more complex instructions are built Fig. 14.1
 - each execution phase (e.g., fetch) consists of one or more sequential micro-ops
 - each micro-op executed in one clock cycle in some subsection of the processor circuitry
 - each micro-op specifies what happens in some area of cpu circuitry
 - cycle time determined by the longest micro-op!
- Micro-ops for (different) instructions can be executed simultaneously
 - non-conflicting, independent areas of circuitry

11/10/2000 Copyright Teemu Kerola 2000 3

Instruction Fetch Cycle ⁽¹⁰⁾ Fig. 11.7

- 4 registers involved
 - MAR, MBR, PC, IR
- What happens?
 - Address of next instruction is in PC
 - Address (MAR) is placed on address bus
 - READ command given to memory
 - Result (from memory) appears on data bus
 - Data from data bus copied into MBR
 - PC incremented by 1
 - New instruction moved from MBR to IR
 - MBR available for new work

11/10/2000 Copyright Teemu Kerola 2000 4

Instruction Fetch Micro-ops ⁽³⁾

- 4 micro-ops
 - can not change order
 - s2 must be done after s1
 - s3 can be done simultaneously with s2
 - s4 can be done with s3, but must implicit

s1: MAR ← (PC), READ
s2: MBR ← (mem)
s3: PC ← (PC) + 1
s4: IR ← (MBR)

t1: MAR ← (PC), READ
t2: MBR ← (mem)
t3: PC ← (PC) + 1
t3: IR ← (MBR)

⇒ Need 3 ticks:

Assume: mem read in one cycle

11/10/2000 Copyright Teemu Kerola 2000 5

Micro-op Grouping

- Must have proper sequence
 - t1: MAR ← (PC)
 - t2: MBR ← (mem)
- No conflicts
 - no write to/read from with same register (set?) at the same time
 - each circuitry can be used by only one micro-op at a time
 - ALU

t2: MBR ← (mem)
t3: IR ← (MBR)

t2: PC ← (PC) + 1
t3: RI ← (RI) + (MBR)

11/10/2000 Copyright Teemu Kerola 2000 6

Micro-op Types ⁽⁴⁾

- Transfer data from one reg to another
- Transfer data from reg to external area
 - memory
 - I/O
- Transfer data from external to register
- ALU or logical operation between registers

11/10/2000 Copyright Teemu Kerola 2000 7

Indirect Cycle

- Instruction contains indirect address of an operand, instead of direct operand address

11/10/2000 Copyright Teemu Kerola 2000 8

Interrupt Cycle

- After execution cycle test for interrupts
- If interrupt bits on, then
 - save PC to memory
 - jump to interrupt handler
 - or, find out first correct handler for this type of interrupt and then jump to that (need more micro-ops)
 - context saved by interrupt handler

t1: MBR ← (PC)
 t2: MAR ← save-address
 PC ← routine-address
 t3: mem ← (MBR)

implicit - just wait?

11/10/2000 Copyright Teemu Kerola 2000 9

Execute Cycle ⁽³⁾

- Different for each op-code

ADD R1, X

t1: MAR ← (IR_{address})
 t2: MBR ← (memory)
 t3: R1 ← (R1) + (MBR)

ADD R1, R2, R3

t1: R1 ← (R2) + (R3)

JMP LOOP

t1: PC ← (IR_{address})

BZER R1, LOOP

t1: if ((R1)=0) then
 PC ← (IR_{address})

Was this updated in indirect cycle?

Can this be done in one cycle?

11/10/2000 Copyright Teemu Kerola 2000 10

Execute Cycle (contd) ⁽¹⁾

t1: MAR ← (IR_{address})
 MBR ← (PC)
 t2: PC ← (IR_{address})
 memory ← (MBR)
 t3: PC ← (PC) + 1

11/10/2000 Copyright Teemu Kerola 2000 11

Instruction Cycle ⁽³⁾

- Decomposed to micro-ops
- State machine for processor
 - state: execution phase Fig. 14.3
 - sub-state: current group of micro-ops
- In each sub-state the control signals have specific values dependent
 - on that sub-state
 - on IR register fields and flags Fig. 14.4
 - including control signals from the bus
 - including values (flags) produced by previous sub-state

11/10/2000 Copyright Teemu Kerola 2000 12

Control State Machine ⁽²⁾

- Each state defines current control signal values Control execution
 - determines what happens in next clock cycle
- Current state and current register/flag values determine next state Control sequencing

11/10/2000 Copyright Teemu Kerola 2000 13

Control Signal Types ⁽³⁾

- Control data flow from one register to another
- Control signals to ALU
 - ALU does also all logical ops
- Control signals to memory or I/O devices
 - via control bus

11/10/2000 Copyright Teemu Kerola 2000 14

Control Signal Example ⁽⁴⁾

- Accumulator architecture Fig. 14.5
- Control signals for given micro-ops cause micro-ops to be executed Table 14.1
 - setting C_2 makes value stored in PC to be copied to MAR in next clock cycle
 - C_2 controls Input Data Strobe for MAR (see Fig. A.30 for register circuit)
 - setting C_R & C_5 makes memory perform a READ and value in data bus copied to MBR in next clock cycle

11/10/2000 Copyright Teemu Kerola 2000 15

Example: Intel 8085 ⁽⁵⁾

- Introduced 1976
- 3, 5, or 6 MHz, no cache
- 8 bit data bus, 16 bit address bus
 - multiplexed
- One 8-bit accumulator

LDA MyNumber	opcode address 0x3A 0x10A5	3 bytes
OUT #2	opcode port 0x2B 0x02	2 bytes

11/10/2000 Copyright Teemu Kerola 2000 16

Example: i8085 ⁽⁶⁾

- Instead of complex data path all data within CPU go via internal bus Fig. 14.7
 - may not be good approach for superscalar pipelined processor - bus should not be bottleneck Table 14.2
- External signals
- Each instruction is 1-5 machine cycles
 - one external bus access per machine cycle
- Each machine cycle is 3-5 states
- Each state is one clock cycle Fig. 14.9
- Example: OUT instruction

11/10/2000 Copyright Teemu Kerola 2000 17

Hardwired Control Logic Implementation ⁽³⁾

Initial representation:

Sequencing control:

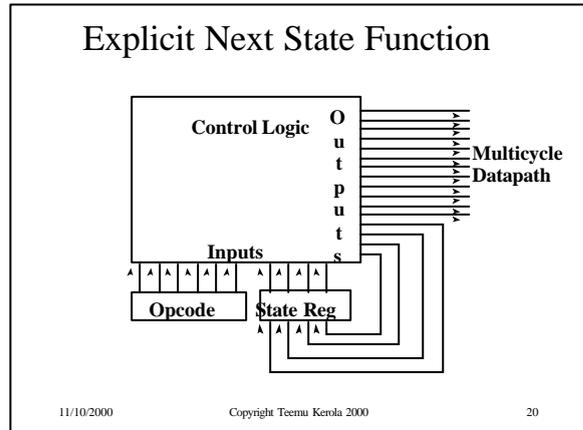
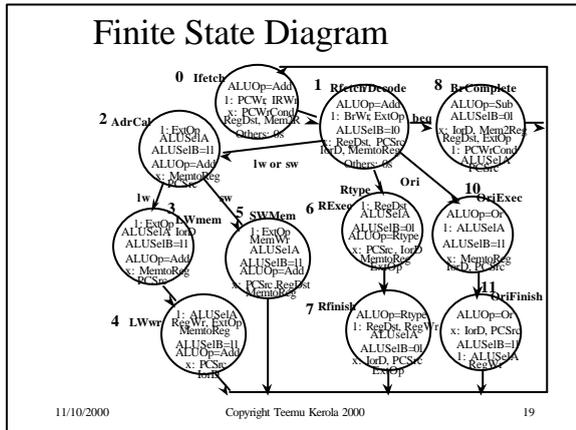
Logic representation:

Implementation:

```

graph TD
    A((Finite state diagram)) --> B((Explicit next state function))
    B --> C((Logic equations))
    C --> D((Programmable Logic Array))
    D --> E((PLA))
            
```

11/10/2000 Copyright Teemu Kerola 2000 18



Logic Equations

Next state from current state	Alternatively, prior state & condition
- State 0 -> <u>State 1</u>	S4, S5, S7, S8, S9, S11 -> State 0
- State 1 -> S2, S6, S8, S10	-> State 1
- State 2 -> _____	-> State 2
- State 3 -> _____	-> State 3
- State 4 -> <u>State 0</u>	-> State 4
- State 5 -> <u>State 0</u>	State 2 & op = SW -> State 5
- State 6 -> <u>State 7</u>	-> State 6
- State 7 -> <u>State 0</u>	State 6 -> State 7
- State 8 -> <u>State 0</u>	-> State 8
- State 9 -> <u>State 0</u>	State 2 & op = JMP -> State 9
- State 10 -> <u>State 11</u>	-> State 10
- State 11 -> <u>State 0</u>	State 10 -> State 11

11/10/2000 Copyright Teemu Kerola 2000 21

- ### Hardwired Control Logic ⁽³⁾
- Circuitry becomes very big and complex very soon
 - may be unnecessarily slow
 - simpler is smaller, and thus faster
 - Many lines (states) exactly or almost similar
 - Have methods to find similar lines and combine them
 - not simple
 - save space, may lose in speed
- 11/10/2000 Copyright Teemu Kerola 2000 22

