Hardwired Control Unit
Ch 14

Micro-operations
Controlling Execution
Hardwired Control

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

Micro-operations (2)

(mikro-operaatio)

- 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

Instruction Fetch Cycle (10)

- 4 registers involved   Fig. 11.7
  - MAR, MBR, PC, IR
- What happens?

	micro-ops?
Address of next instruction is in PC	
Address (MAR) is placed on address bus	MAR ← (PC)
READ command given to memory	READ
Result (from memory) appears on data bus	
Data from data bus copied into MBR	MBR ← (mem)
PC incremented by 1	PC ← (PC) +1
New instruction moved from MBR to IR	IR ← (MBR)
MBR available for new work	

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 be done after s2

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

⇒ Need 3 ticks:

implicit

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

Assume: mem read in one cycle

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

t2:	MBR ← (mem)
t3:	IR ← (MBR)

  - each circuitry can be used by only one micro-op at a time
    - E.g., ALU

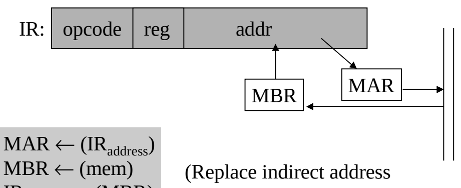
t2:	PC ← (PC) +1
t3:	R1 ← (R1) + (MBR)

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

Indirect Cycle

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



$MAR \leftarrow (IR_{address})$
$MBR \leftarrow (mem)$
$IR_{address} \leftarrow (MBR)$

(Replace indirect address by direct address)

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: $MBR \leftarrow (PC)$
t2: $MAR \leftarrow$ save-address
$PC \leftarrow$ routine-address
t3: $mem \leftarrow (MBR)$

implicit - just wait?

Execute Cycle (3)

- Different for each op-code

ADD R1, X
t1: $MAR \leftarrow (IR_{address})$
t2: $MBR \leftarrow (memory)$
t3: $R1 \leftarrow (R1) + (MBR)$

ADD R1, R2, R3
t1: $R1 \leftarrow (R2) + (R3)$

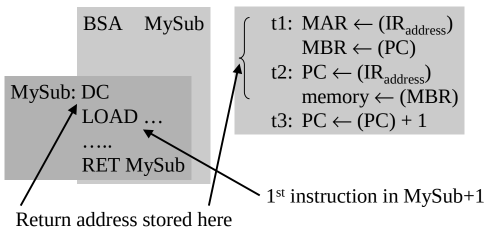
JMP LOOP
t1: $PC \leftarrow (IR_{address})$

Was this updated in indirect cycle?

BZER R1, LOOP
t1: if ((R1)=0) then
$PC \leftarrow (IR_{address})$

Can this be done in one cycle?

Execute Cycle (contd) (1)



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

Instruction Cycle (3)

- Decomposed to micro-ops
- State machine for processor
  - state: execution phase
  - sub-state: current group of micro-ops

Fig. 14.3

- In each sub-state the control signals have specific values dependent
  - on that sub-state
  - on IR register fields and flags
    - including control signals from the bus
    - including values (flags) produced by previous sub-state

Fig. 14.4

Control State Machine (2)

- Each state defines current control signal values
  - determines what happens in next clock cycle

Control execution

- Current state and current register/flag values determine next state

Control sequencing

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

Control Signal Example (4)

- Accumulator architecture Fig. 14.5
- Control signals for given micro-ops <u>cause</u> 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

Example: Intel 8085 (5)

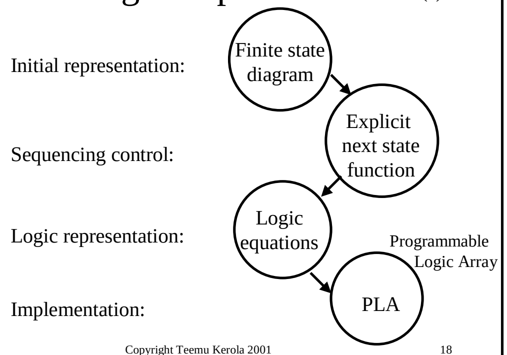
- Introduced 1976
- 3, 5, or 6 MHz, no cache Fig. 14.7
- 8 bit data bus, 16 bit address bus
  - multiplexed
- One 8-bit accumulator

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

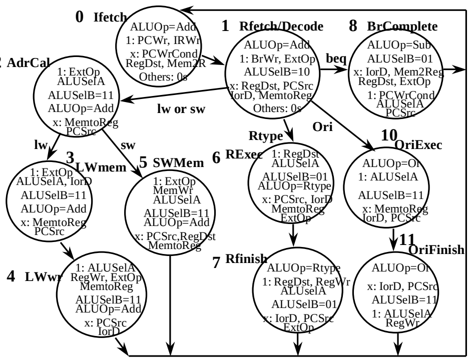
Example: i8085 (6)

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

Hardwired Control Logic Implementation (3)



Finite State Diagram



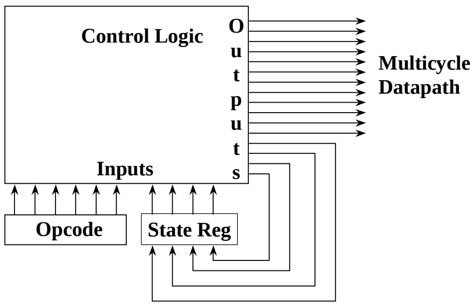
10/10/2001 19

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 (states) and combine them
  - not simple
  - save space, may lose in speed

10/10/2001 22

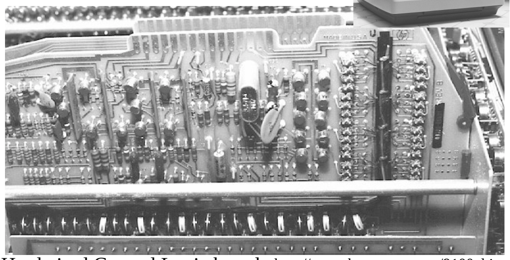
Explicit Next State Function



10/10/2001 20

-- End of Chapter 14: Hardwired Control --

HP 9100 Calculator (1968), 20 kg,
$5000, 16 regs (data or 14 instructions/reg),
32Kb ROM, 2208 bit RAM magnetic core memory



Hardwired Control Logic board http://www.hpmuseum.org/9100cl.jpg

10/10/2001 23

Logic Equations

Next state from current state

- State 0 -> State1
- State 1 -> S2, S6, S8, S10
- State 2 ->__________
- State 3 ->__________
- State 4 ->State 0
- State 5 -> State 0
- State 6 -> State 7
- State 7 -> State 0
- State 8 -> State 0
- State 9-> State 0
- State 10 -> State 11
- State 11 -> State 0

Alternatively,
prior state & condition

S4, S5, S7, S8, S9, S11	-> State0
_________________	-> State1
_________________	-> State 2
_________________	-> State 3
_________________	-> State 4
State2 & op = SW	-> State 5
_________________	-> State 6
State 6	-> State 7
_________________	-> State 8
State3 & op = JMP	-> State 9
_________________	-> State 10
State 10	-> State 11

10/10/2001 21