

Microprogrammed Control

CIT 595
Spring 2008

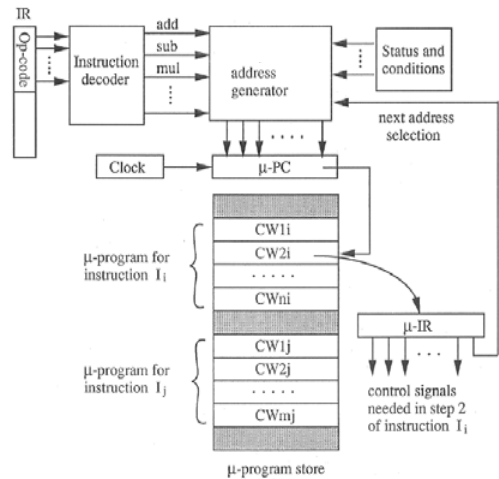
Microprogrammed Control

- Each machine instruction is in turn implemented by a series of instructions called *microinstructions*
 - A microinstruction encodes
 - Control signal for carry out a particular stage in the instruction cycle
 - Next (likely) microinstruction
- The microinstructions form a *microprogram*, which is stored in a memory
 - Sometimes called Control Store
 - Memory is non-volatile and reprogrammable e.g. EPROM, Flash
- Microprogram Control is essentially a Finite State Machine (programmable)

CIT 595

2

μ-Programmed Control

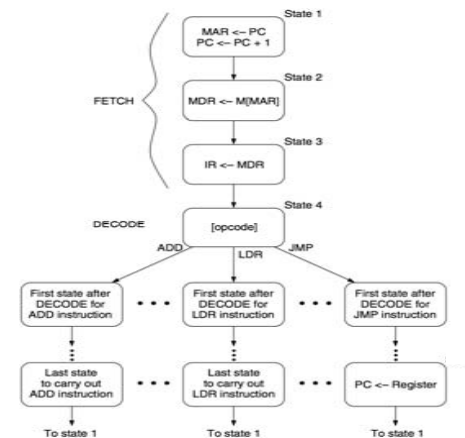


Source:
<http://fourier.eng.hmc.edu/~processor/node11.html>

CIT 595

3

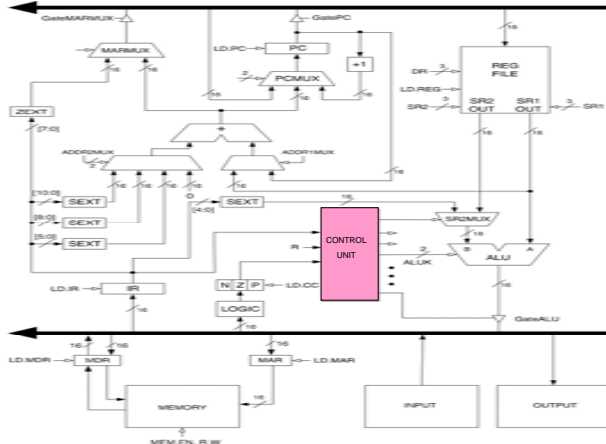
Microprogrammed Control: State Diagram



CIT 595

4

Another Implementation of LC3

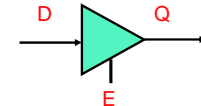


CIT 595 Filled arrow = info to be processed. Unfilled arrow = control signal.

5

Some Implementation detail : Common Bus

- This style of implementation uses a common bus
 - Shared by memory, ALU, Register File, I/O
 - Same set of wires carry address and data information
- To have only one component's address/data on the bus
 - Use device called Tristate
 - $E = 1, D = Q$
 - $E = 0$, no connection between D & Q
 - Tristate is an alternative to mux
- In hardwired implementation we had point-to-point connection between components
 - Buses can be of various kinds
 - More on Bus Architectures later



CIT 595

6

Some Implementation detail : Memory

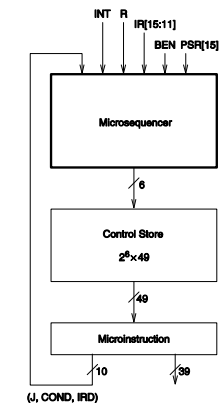
- Unified Memory for instruction and data
- Before Address or Data is given to memory, there is temporary register to receive the data from the common bus
 - MAR = Memory Address Register
 - MDR = Memory Data Register
- This is done as the bus is used for both address and data
 - E.g. STR you need to provide address and data
- The register can be Read or Written by control signals
 - LD.MAR for (MAR) and LD.MDR for (MDR)
- Similarly when instruction is fetched from Memory, the content is placed in register called IR and it controlled by LD.IR

CIT 595

7

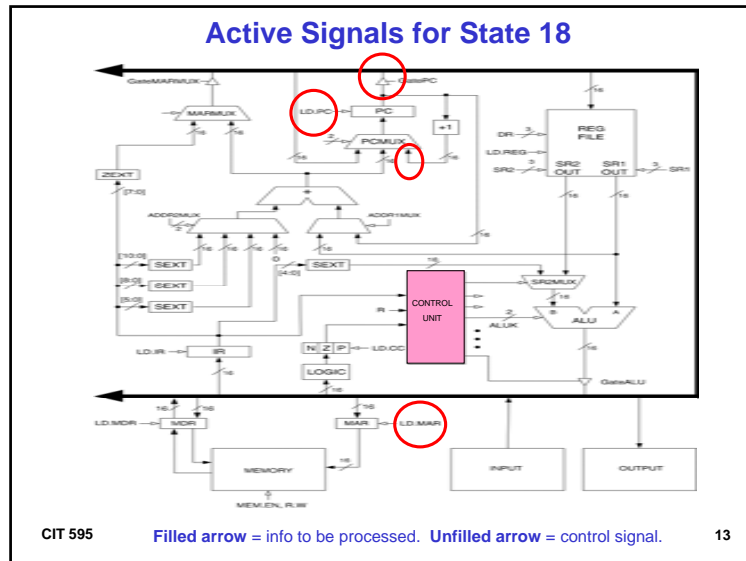
LC3 Implementation using Microprogram Control

- The behavior of LC-3 during a given clock cycle is completely described by the 49 bit microinstruction
 - 39 bits for control signals
 - 10 bits for possible next state of the machine
- Each phase of instruction cycle may require more than one microinstruction
 - E.g. Fetch stage takes 3 microinstructions
- 6-bit address is used to lookup memory
 - There are 52 possible microinstructions (states) that can describe LC3's behavior
 - Memory size $2^6 \times 49$



CIT 595

8



LC3 Microprogram Implementation: Next State

- 10 bits of current microinstruction
 - J (6-bits): encodes the mostly likely next state
- COND (3-bits): field indicates special tests that must occur to compute the true next state
 - 0 – Unconditional (no test to be performed - then next state is J)
 - 1 – Memory Ready
 - 2 – Branch
 - 3 – Addressing Mode
 - 4 – Privilege Mode
 - 5 – Interrupt
- IRD (1-bit) : If IRD = 1, ignore J and COND
 - Next state is based on opcode field(15:12)
 - This only happens in state 32

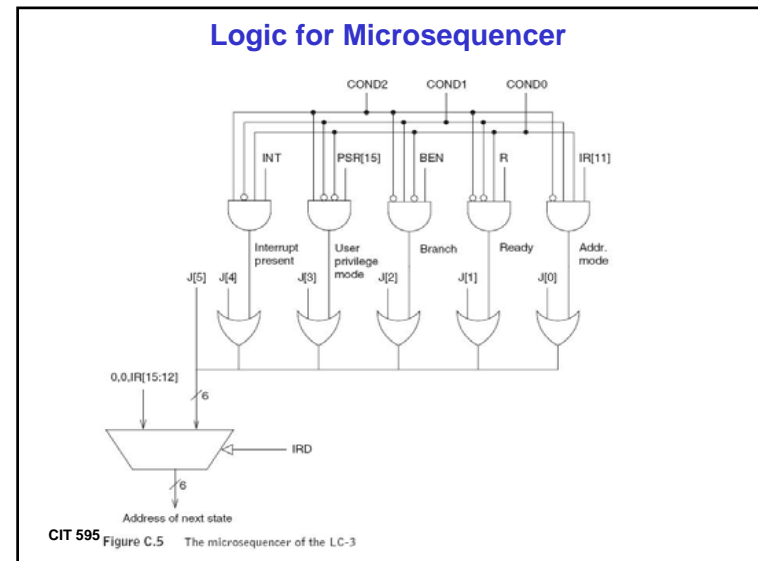
CIT 595 14

LC3 Microprogram: Next State (contd..)

9 other signals (what actual event took place)

- INT: To indicate interrupt from I/O device
 - Only tested if in state 18 (because that is before the start of an instruction cycle)
- R: indicate the end of memory operation
- IR[15:12]: current opcode (instruction bits 15:12)
- IR[11]: Addressing mode (user subroutine)
- PSR[15]: processor executing in supervisor or user mode
- BEN: indicates whether or not a branch should be taken

CIT 595 15



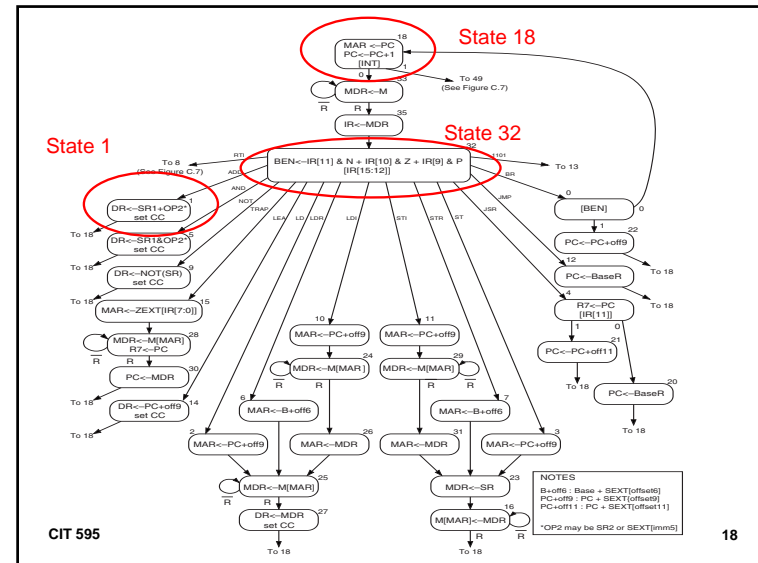
LC3 Next State Example

CURR STATE	J	COND	IRD	POSSIBLE NEXT STATE
18	33	5	0	33,49
1	18	0	0	18
32	0	0	1	0-15

- State 18: Most likely next state is 33 but need to check for INT (COND = 5). If INT = 1, Next State = 49
- State 1: Next State is just 18 (this because you are done finishing ADD instr and want to start a new instr. cycle)
- State 32: J and COND ignored as IRD = 1, 0 – 15 are your next possible states

CIT 595

17



CIT 595

18

Hardwired vs. Microprogrammed

- **Complexity**
 - There is an extra level of instruction interpretation in microprogrammed control, which makes it slower than hardwired control
- **Flexibility**
 - Instruction and Control Logic are tied together in hardwired control, which makes it difficult to modify
 - New instructions can be easily added by only making changes to the microprogram in microprogrammed control implementation

CIT 595

19