Computer Organization & Architecture

“Computer Organization and Embedded Systems”

Carl Hamacher, Zvonko Vranesic, Safwat Zaky, Naraig Manjikian

Control signals

- Ex., RF_write signal is set to 1 in step T5 during execution of instruction that writes data into register file
- May be generated by the logic expression

  $$RF\_write = T5 \cdot (ALU + Load + Call)$$

  - ALU stands for all instructions that perform arithmetic or logic operations
  - Load stands for all Load instructions
  - Call stands for all subroutine-call and software-interrupt instructions
- RF_write signal is a function of both instruction and timing signals

Control signals

- Setting of some of multiplexers need not change from one timing step to another
  - Multiplexer’s select signal can be implemented as a function of instruction only. For example,

    B_select = Immediate

  where:
  - Immediate stands for all instructions that use an immediate value in the IR

Control signals

- If MEM_read or a MEM_write command is issued:
  - Does not end until the MFC signal is asserted
- To extend duration of execution step to more than one clock cycle:
  - Need to disable the step counter
- Assume counter is incremented when enabled by a control signal called Counter_enable:
  - Let need to wait for memory operation to be completed be indicated by a control signal called WMFC
- Counter_enable should be set to 1 in any step where WMFC is not asserted
  - Otherwise, it should be set to 1 when MFC is asserted

$$\textbf{Counter\_enable} = \overline{\textbf{WMFC}} + \textbf{MFC}$$

Control signals

- New value is loaded into PC at the end of any clock cycle in which PC_enable signal is activated.
  - Must ensure that PC is incremented only once when an execution step is extended for more than one clock cycle
  - When fetching an instruction, the PC should be enabled only when MFC is received.
- PC is also enabled in step 3 of branch instructions
  - Let BR denote all instructions in this group
- PC_enable may be realized as

$$\mathbf{PC\_enable = T1 \cdot MFC + T3 \cdot BR}$$

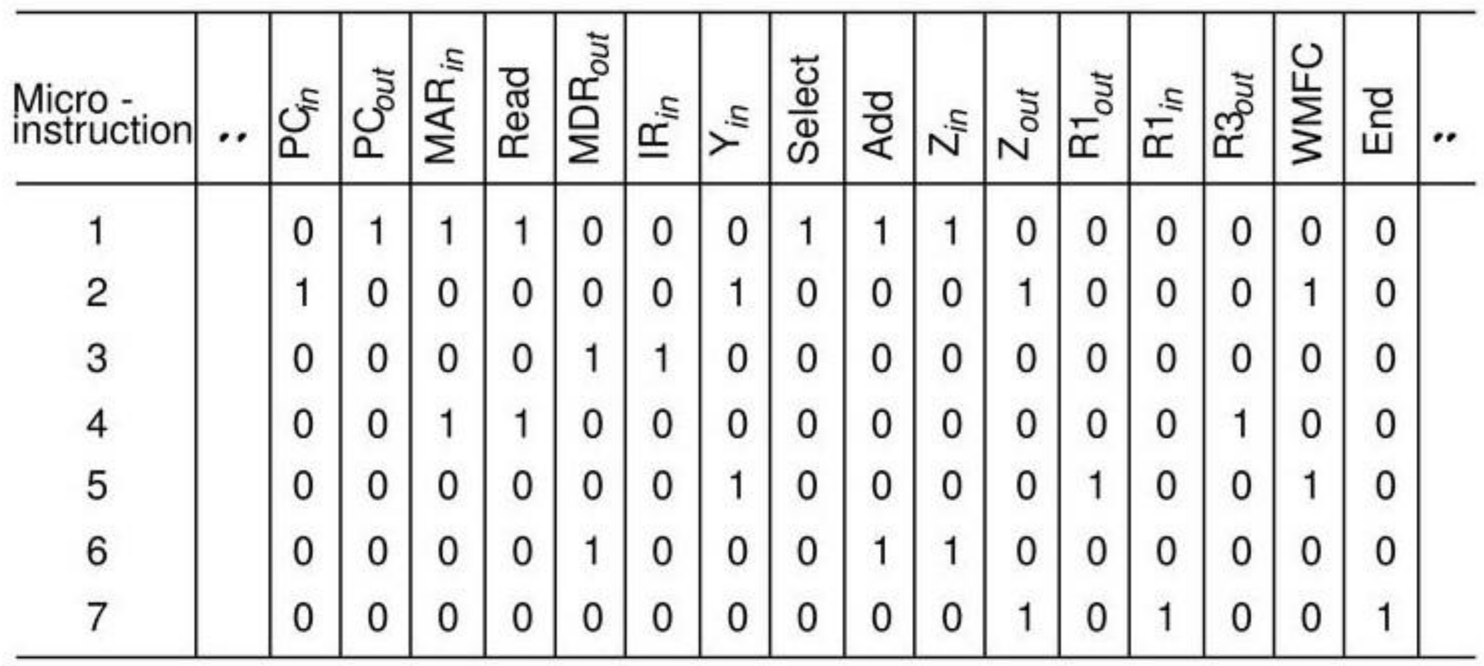
Microprogrammed control

- Control signals are generated for each execution step based on instruction in IR
- In hardwired control, signals are generated by circuits that interpret
  - Contents of IR
  - Timing signals derived from a step counter
- Instead of circuits, it is possible to use “software" approach
  - Desired setting of the control signals in each step is determined by a program stored in a special memory

Microprogrammed control

- Control program is called a **microprogram**
  - Stored on processor chip in a small and fast memory called **control store**
- **Control word** (CW): word whose each bit represent
  - Control signals for each step in sequence
- **Microroutine**: sequence of control words
  - Corresponding to control sequence for a machine instruction
  - Individual control words are referred to as **microinstructions**
  - Microroutines for all machine instructions are stored in control store

Microprogrammed control



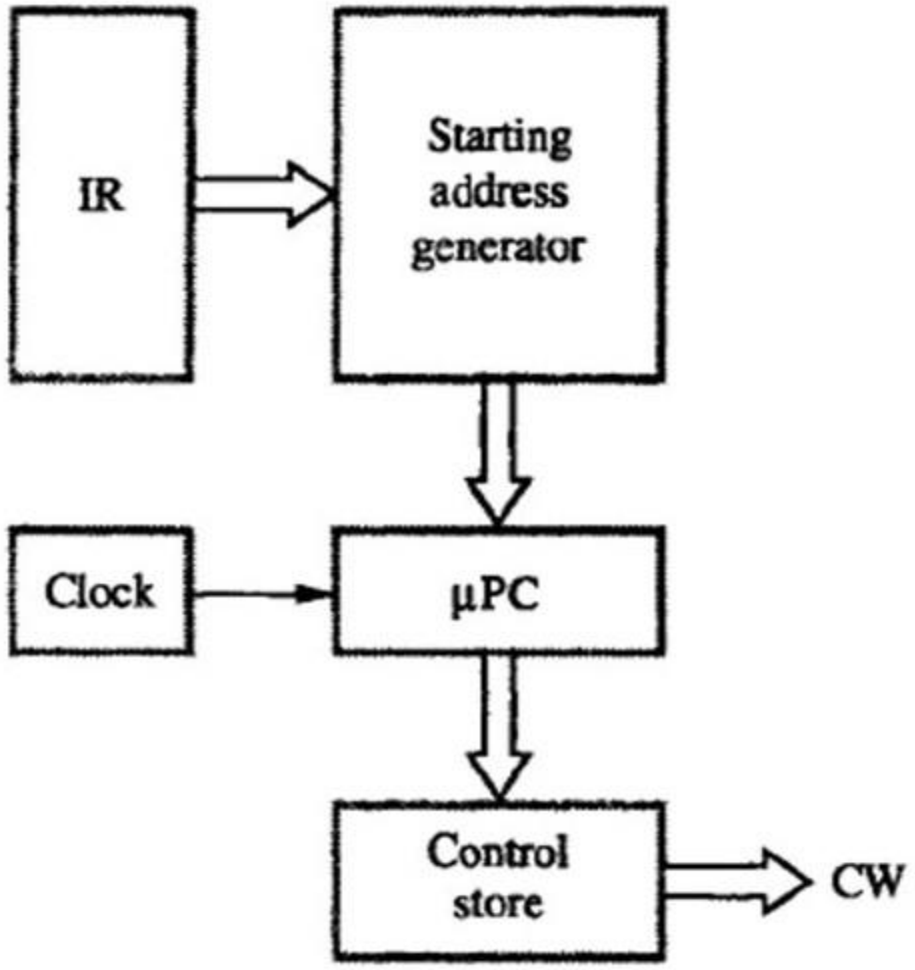
Micro - instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	
4		0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	
5		0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	
6		0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	
7		0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	

Microprogrammed control

- Suppose *n* control signals are needed
- Let each control signal be represented by a bit in *n*-bit word
  - Often referred to as a control word or a microinstruction
- Each bit in that word specifies:
  - Setting of corresponding signal for a particular step in execution flow
- Ex., action of reading an instruction or a data operand from memory requires use of MEM_read and WMFC signals
  - Signals are asserted by setting corresponding bits in control word in required steps
- When a microinstruction is read from the control store:
  - Each control signal takes on the value of its corresponding bit

Microprogrammed control

- Control unit can generate control signals:
  - By sequentially reading CWs of corresponding microroutine
  - To read CWs sequentially, micro-program counter (µPC) is used
  - Every time new instruction is loaded in IR, appropriate starting address is loaded in µPC
  - µPC incremented at every clock cycle


IR
Starting address generator
Clock
μPC
Control store
CW

Microinstructions

- Assign one bit to each control signal
  - Long microinstructions
  - At any given time, only few bits are set
- Assume:
  - 42 control signals are required: for general purpose registers, ALU operations, memory red/write, I/O read/write, etc.
  - Most signal are not needed simultaneously
  - Group mutually-exclusive signals together

Microinstruction

F1	F2	F3	F4	F5

F1 (4 bits)	F2 (3 bits)	F3 (3 bits)	F4 (4 bits)	F5 (2 bits)
0000: No transfer	000: No transfer	000: No transfer	0000: Add	00: No action
0001: PC $_{out}$	001: PC $_{in}$	001: MAR $_{in}$	0001: Sub	01: Read
0010: MDR $_{out}$	010: IR $_{in}$	010: MDR $_{in}$	⋮	10: Write
0011: Z $_{out}$	011: Z $_{in}$	011: TEMP $_{in}$		
0100: R0 $_{out}$	100: R0 $_{in}$	100: Y $_{in}$	$\underbrace{\text{1111: XOR}}_{\text{16 ALU functions}}$	
0101: R1 $_{out}$	101: R1 $_{in}$			
0110: R2 $_{out}$	110: R2 $_{in}$			
0111: R3 $_{out}$	111: R3 $_{in}$			
1010: TEMP $_{out}$				
1011: Offset $_{out}$				

F6	F7	F8	• • •

F6 (1 bit)	F7 (1 bit)	F8 (1 bit)
0: SelectY	0: No action	0: Continue
1: Select4	1: WMFC	1: End

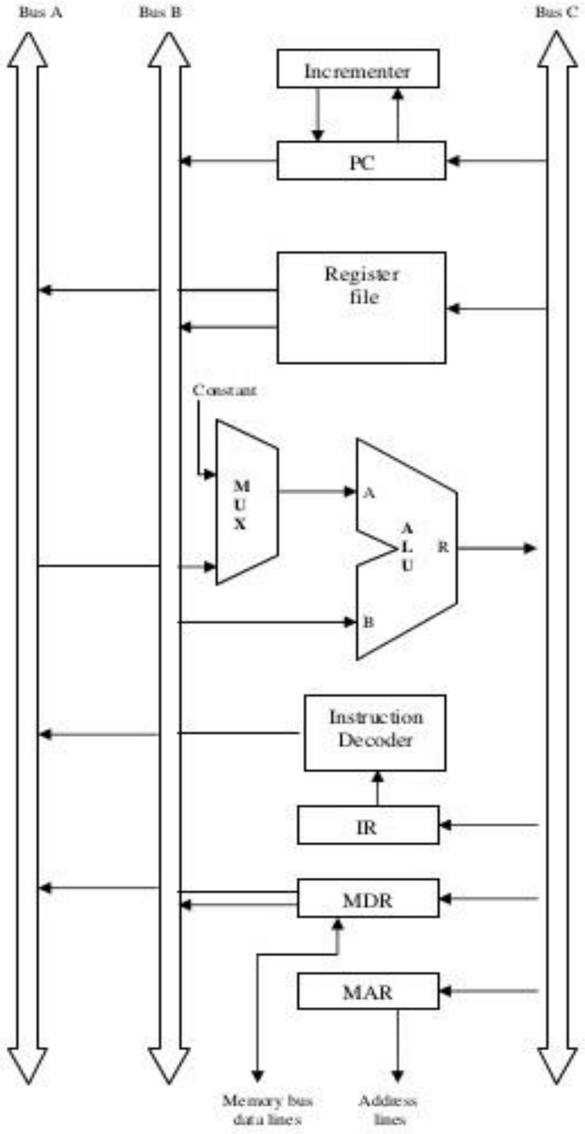
Microinstructions

- **Vertical organization**:
  - Highly encoded schemes using compact codes
  - Specify small number of control signals in each microinstruction
  - Slower operating speeds
  - Fewer bits are required
  - Less hardware needed to execute instructions.
- **Horizontal organization**:
  - Minimally encoded schemes
  - Many resources can be controlled with a single microinstruction
  - Higher operations speed is desired
  - Parallel use of resources is allowed

CISC style processors

- CISC-style instruction sets are more complex:
  - Allow much greater flexibility in accessing instruction operands
  - Unlike RISC-style instruction sets, where only Load and Store instructions access data in memory
  - CISC instructions can operate directly on memory operands
- They are not restricted to one word in length
  - An instruction may use several words to specify operand addresses and actions to be performed

Three-bus processor organization



ADD R5, R6

AND X(R7), R9

END