William Stallings
Computer Organization and Architecture

Chapter 17
Micro-programmed Control

Hardwired vs Micro-programmed Control

- Hardwired implementation of the CU
  - synthesizing a sequential circuit to obtain the desidered input-output relations for control signals
- Micro-programmed implementation of the CU
  - use sequences of micro-operations to implement the execution of CPU instructions
- Called **micro-programming** or **firmware production**, since each sequence is made up by a small number of very simple operations

Implementation (1)

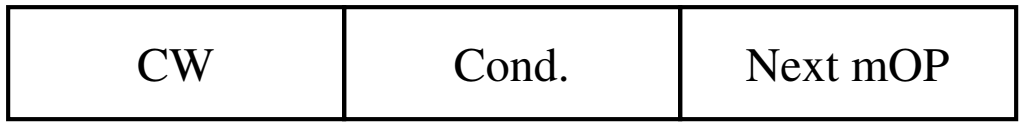
- For each micro-operation (mOP) all the control unit does is to generate a set of control signals
- Each control signal is on or off
- Represent each control signal by a bit
- The set of control bits is a **control word (CW)**
- Each mOP corresponds to a different CW
- Each mOP is executed during one execution cycle of the CU, which starts by reading the current CW to be executed and ends by preparing the address of the next CW to be executed

Implementation (2)

- Example of CWs for the mOPs corresponding to instruction fetch and direct addressing (see 16-26 and 16-27)
  - CW_1: MAR <- PC
    C2
  - CW_2: MBR <- memory; ALU <- PC; increment ALU; AC <- ALU
    C0 CR C5 C14 CA C9
  - CW_3: PC <- AC; IR <- MBR
    C15 C4
  - CW_4: MAR <- $IR_{address}$
    C16
  - CW_5: MBR <- memory
    C0 CR C5
- Add to each CW address information to specify the next mOP, depending on some conditions

Implementation (3)

- Have a sequence of CW for each CPU instruction or substep of it (**micro-procedure**)
- Each micro-procedure is terminated by a (possibly conditional) jump to another micro-procedure



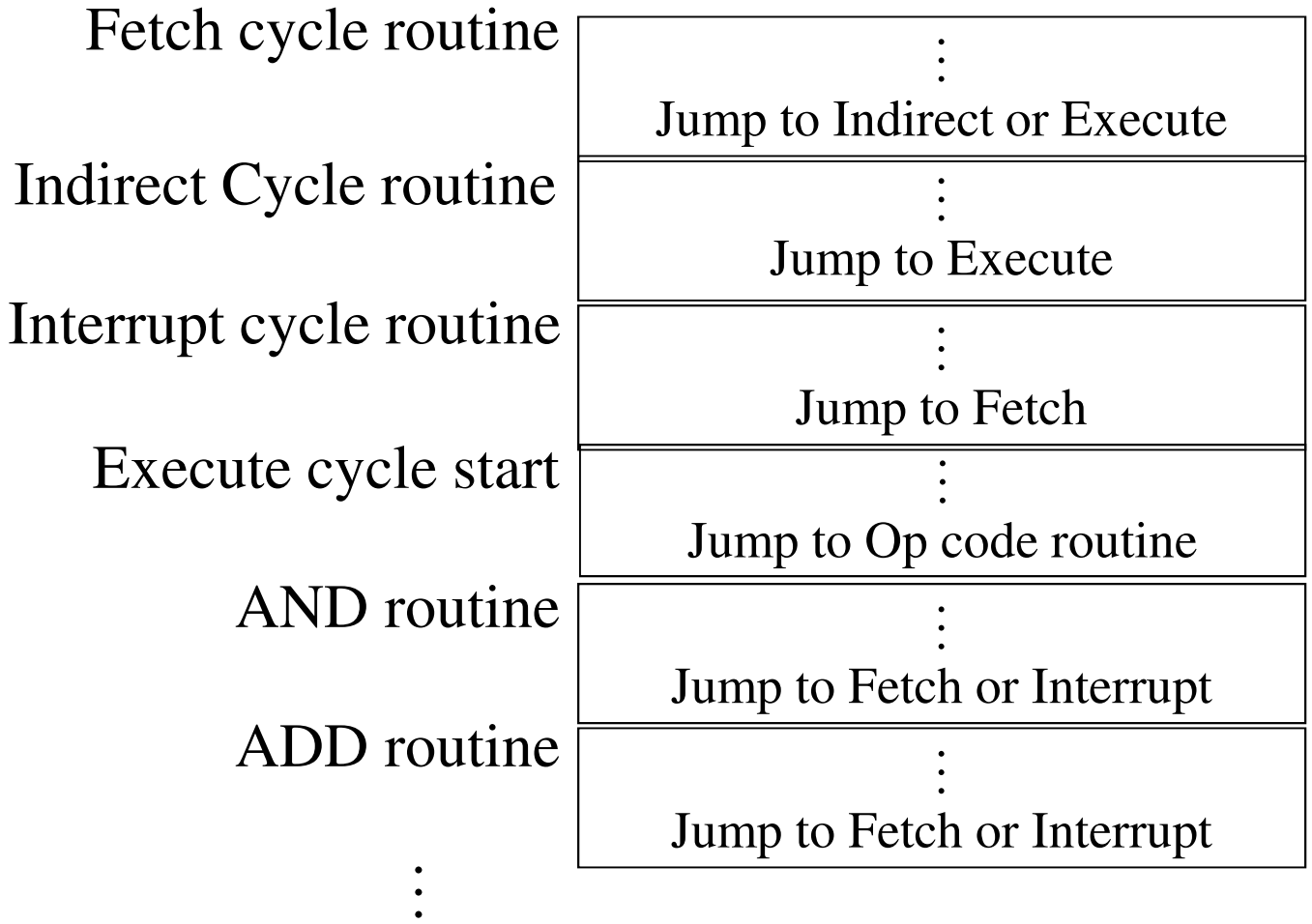
- All CWs are put in a memory, called **Control Memory**, which can now be used to drive the CU behavior
- All is needed is to define the flow of execution of CWs, i.e. the sequence of addresses in the control memory whose corresponding CWs have to be activated

Implementation (4)

curr.CW	mOPs						Jump	Next CW
CW1	C2						*False*	
CW2	C0	C5	C9	C14	CA	CR	*False*	
CW3	C4	C9	C15				?Indirect?	CW6
CW4	C16						*False*	
CW5	C0	C5	CR				*True*	CW10

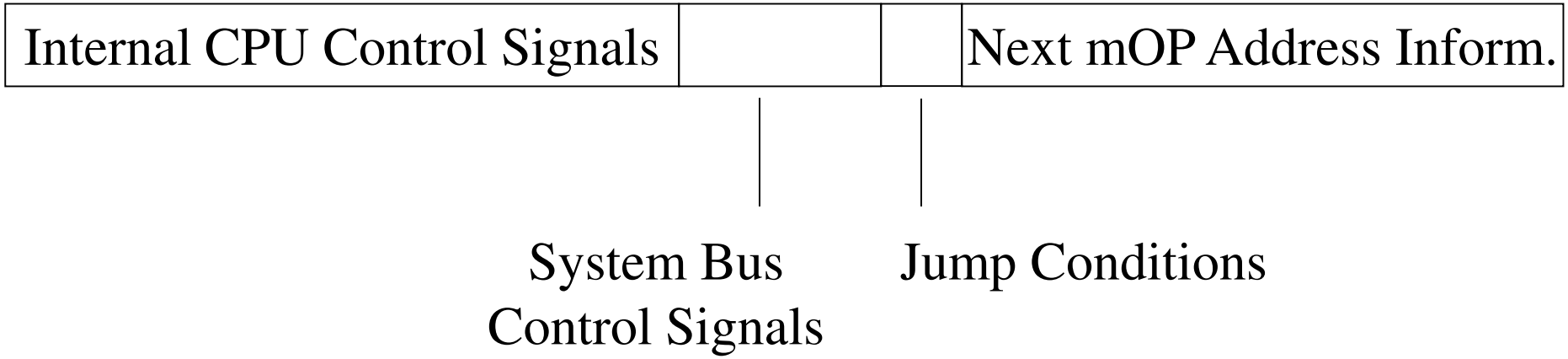
- Assuming the first control word of the micro-procedure for indirect addressing is CW6 and the first one for execute is CW10
- When the (value of the) jump condition is *False* the next CW in the sequence is executed

Control Memory



Horizontal Micro-programming

- Wide CW: reserve one bit of the CW for each control signal
- Many mOPs can be executed in parallel, but a large space is used



Vertical Micro-programming (1)

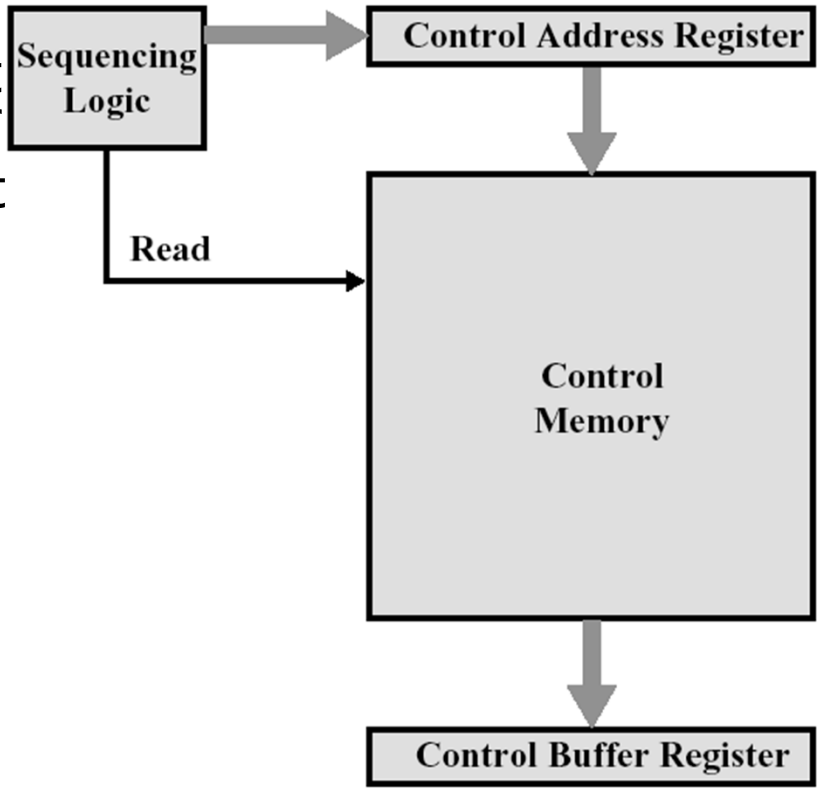
- CW is narrow: *n* control signals encoded into $\log_2 n$ bits
- Limited ability to execute mOP in parallel: at most 1 control signal can be managed
- Encoding of control information requires an additional CW decoder to identify the exact control line being manipulated
- CW decoder introduce a delay

Vertical Micro-programming (2)

- Compromise:
  - Divide control signals into disjoint groups
    - Functional basis (groups for operand source, addressing mode, …)
    - Resource basis (groups for ALU, memory, I/O, …)
  - Criteria
    - All operations coded within a group cannot be executed in parallel
    - Any operation in a group can be executed in parallel with any operation in any other group
  - Implement each group as separate field in memory word
  - Supports reasonable levels of parallelism without too much complexity
  - With *k* groups at most *k* mOPs may be executed in parallel

Control Unit: core elements

- Control Address Register
  - Contains the **address** of the current mOP in execution and (at the end of each CU execution cycle) of the next mOP to be executed
- Control Buffer Register
  - Store the **content** of the current mOP in execution
- Sequencing Logic
  - Activates reading from the Control Memory of the location at the address in CAR and storing its content in CBR
  - Decides on the next address to be put in CAR at the end of the execution cycle



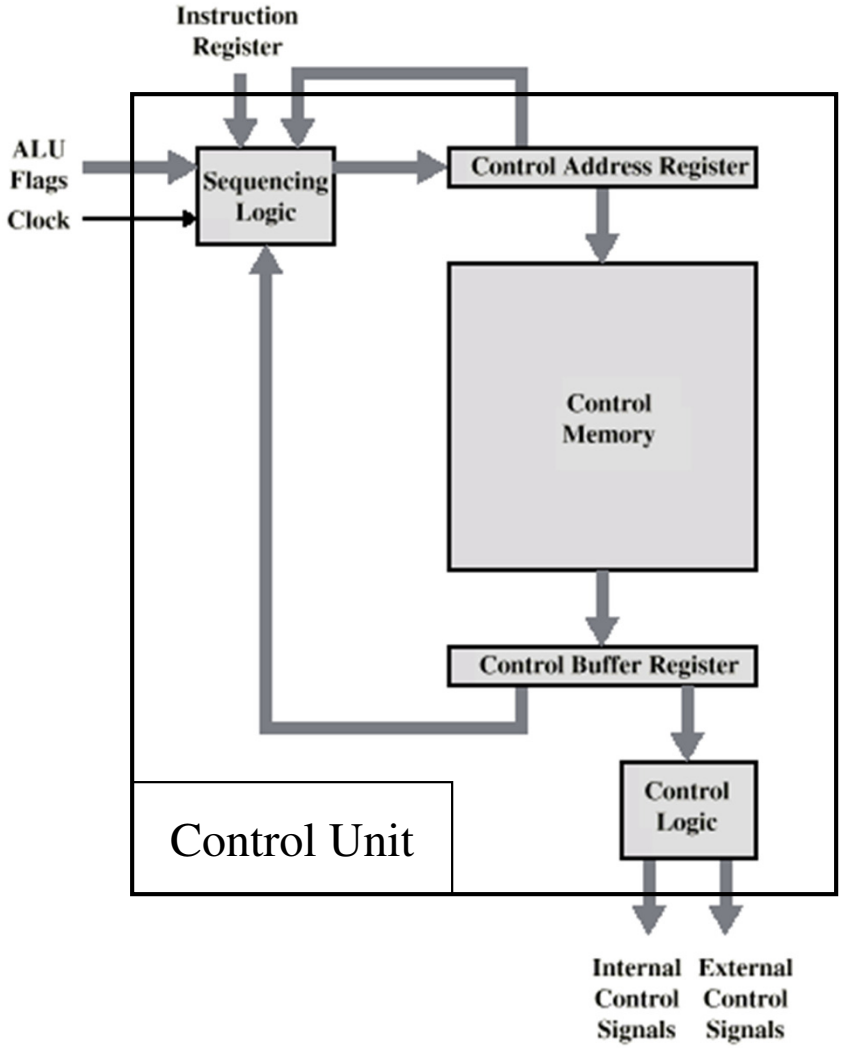
Sequence of operations during each execution cycle of CU (1)

1. Sequencing logic unit issues read command to Control Memory
2. The CW at the address specified in Control Address Register is read into Control Buffer Register
3. CBR content generates control signals to CPU and to system bus, and information used to decide next CW address in the Control Memory

Sequence of operations during each execution cycle of CU (2)

4. Sequencing Logic decide the next CW address based on:
   - jump conditions and next address information in CBR
   - info from IR (D) and from flags (T)
   - current state of the CU, given by the value in CAR
5. then loads the next CW address into the CAR
   - Next CW address in control memory can be
     - Current address + 1
     - A jump to
       - A new micro-procedure within a same CPU instruction
       - A new micro-procedure corresponding to a new CPU instruction

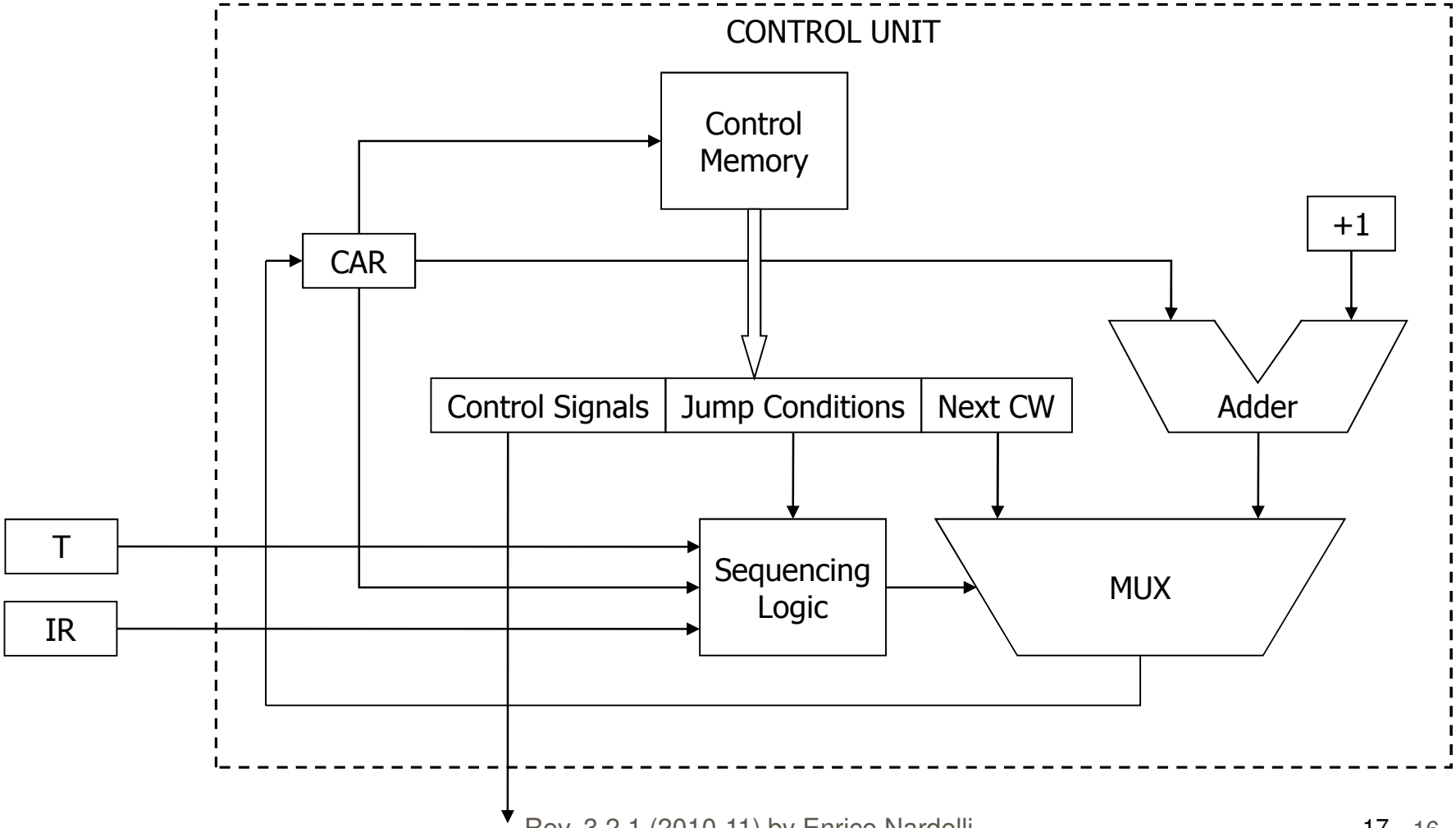
Control Unit Organization



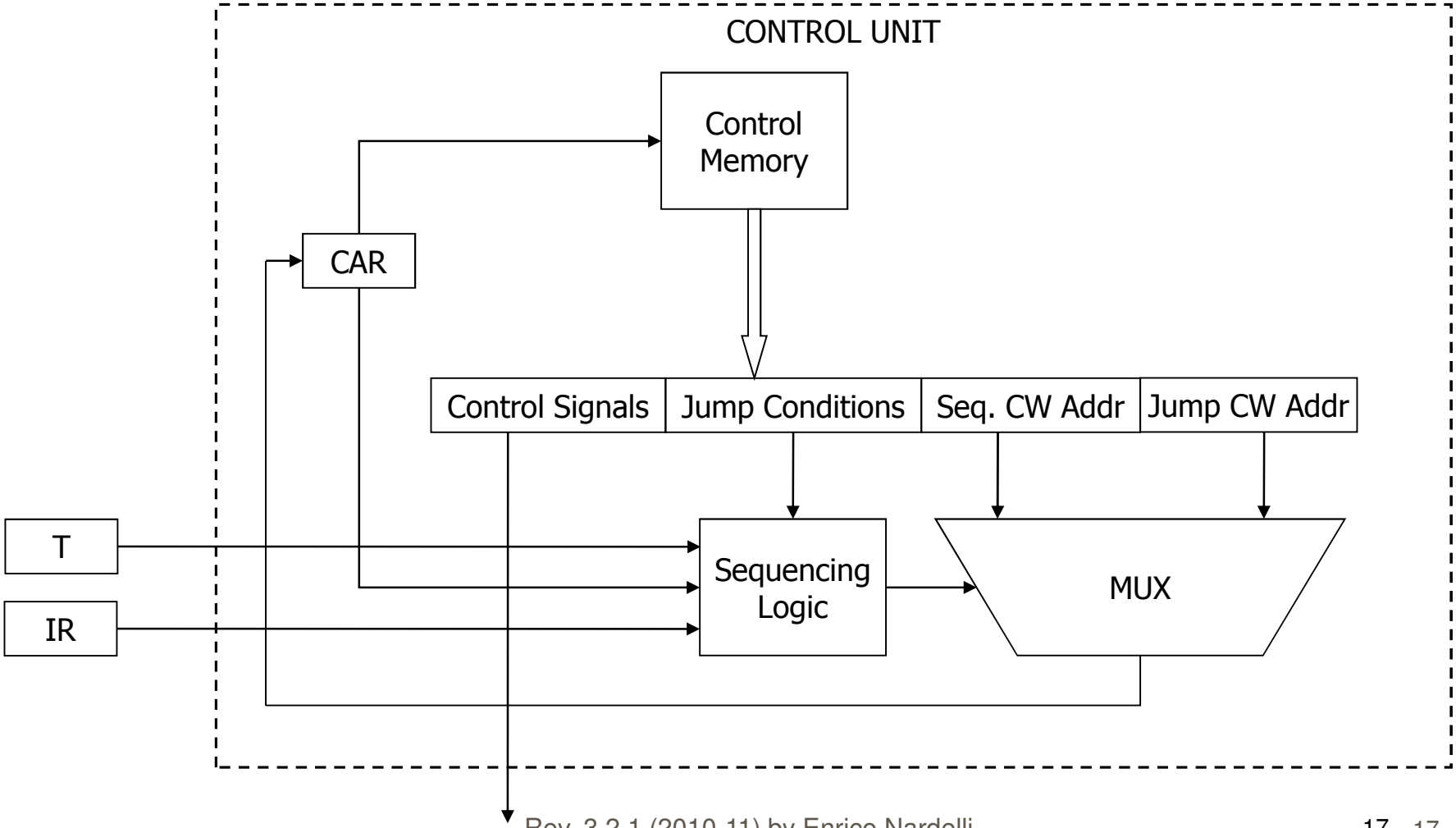
Organization of the sequencing logic

- Sequencing logic decides the address of the next CW to be executed
- Its organization depend also on structure of jump conditions and next-mOP address information in CW
  - 1 field containing only the address in case of jump, since otherwise CU go in sequence (needs an adder)
  - 2 fields containing both addresses needed for the case CU jumps or not (faster but longer CW)
  - Variable structure of CW: only address information or only control information (much shorter CW but slower execution)

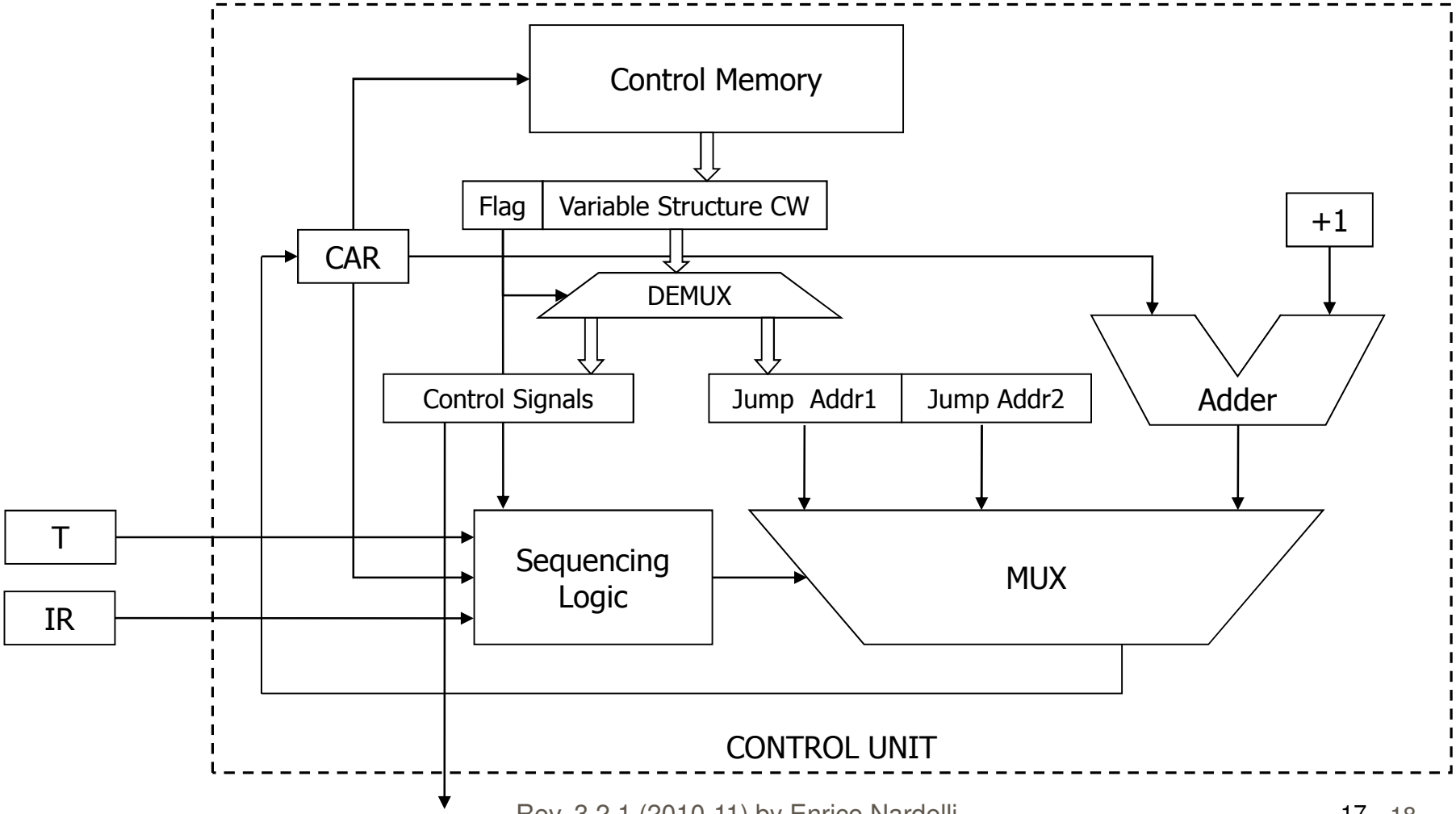
CW structure: 1 next CW field



CW structure: 2 next CW fields



Variable CU structure



Hardwired vs Micro-programmed

- Micro-programmed control simplifies the design of control unit
  - Cheaper
  - Less error-prone
  - Much more easier to revise and modify
- But the control unit is faster with hardwired CU
- Micro-programmed CU is used mainly for CISC architectures since flexibility of CU is more important for a complex instruction set
- On the other side, RISC architectures use hardwired CU since with a simpler instruction set flexibility is a less important requirement than speed of execution