Micro-programmed State Machines

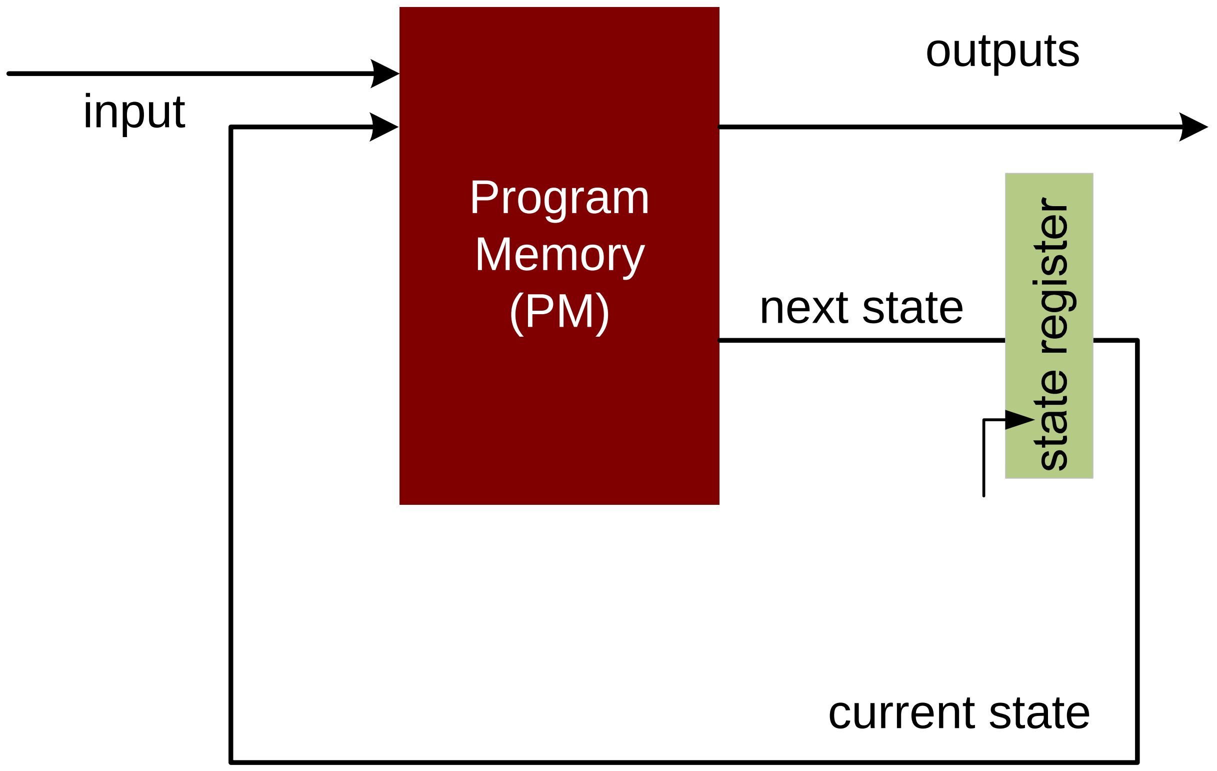
Lecture 10

Dr. Shoab A. Khan

Micro-programmable State machines

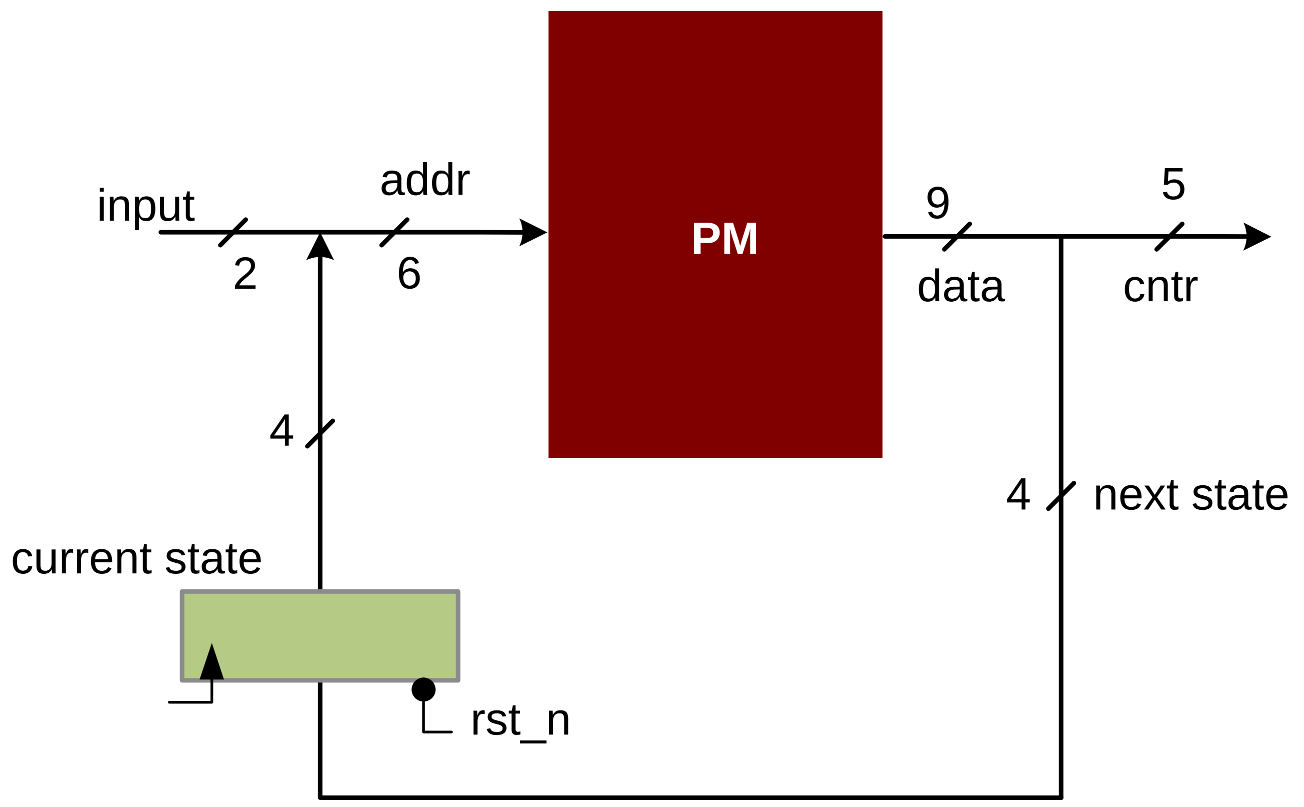
- A Micro-Programmable State machine substitutes combinational cloud of FSM with Programmable Memory (PM)
- The control signals for all the states are stored and read from PM instead of being generated by combination logic

PM-BASED MICRO-PROGRAMMABLE STATE MACHINE



- Mealy machine: PM address bits are function of Input and current states, the PM data bits are outputs and next state

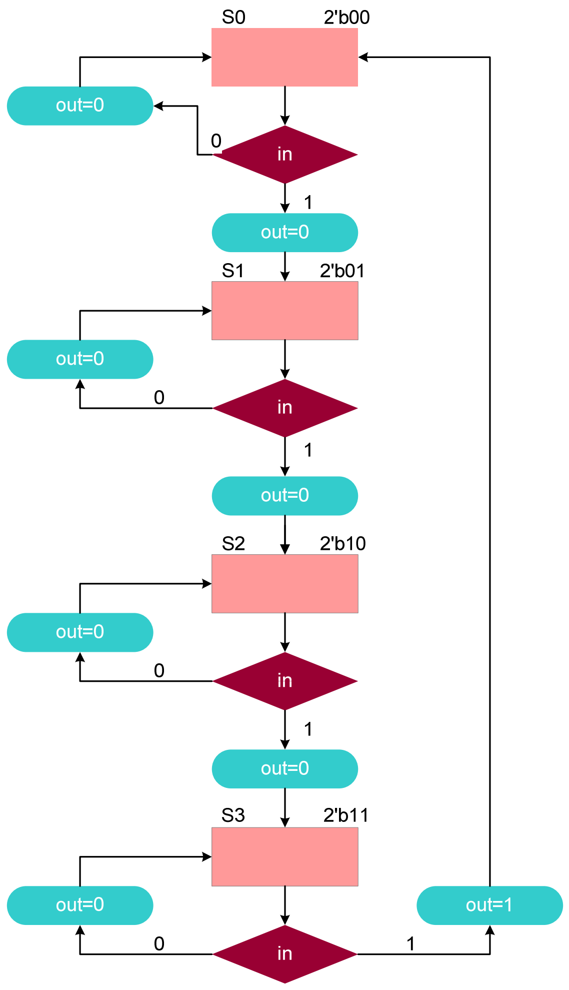
Example Design

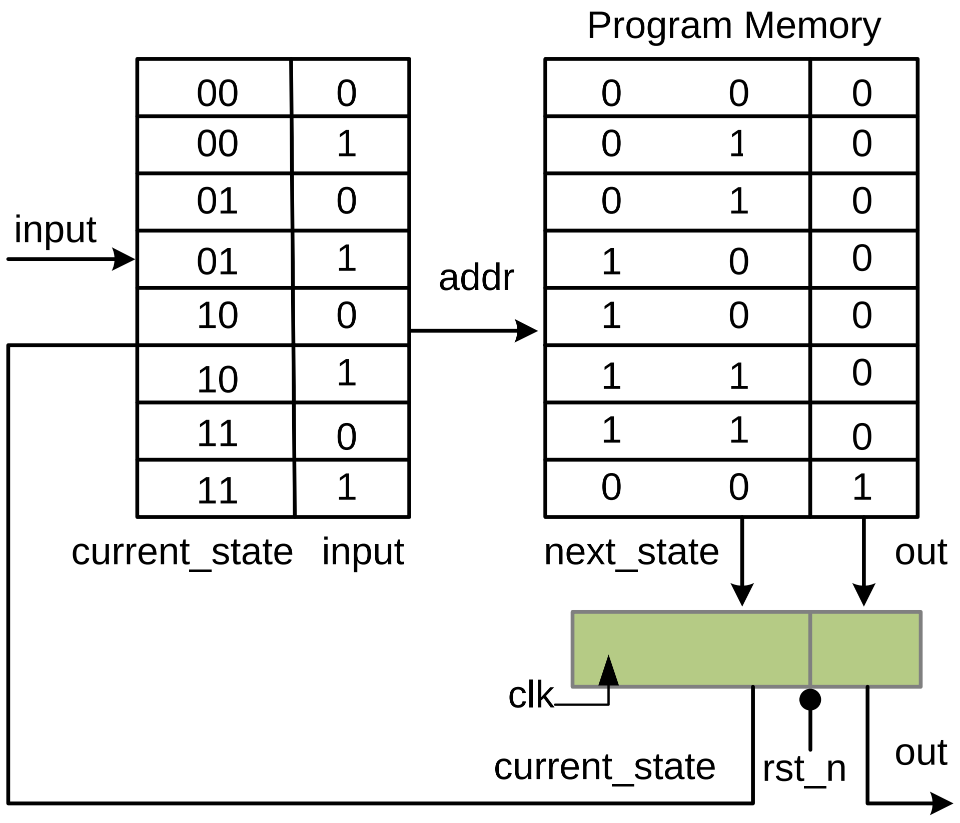


- 2-bit input and 4-bit current state constitutes 6-bit address of PM
- 9-bit wide PM contents are 5-bit output bits for control and 4-bit next state

Micro-programmed Sate Machine Implementation

- Number of 1s detected problem of Chapter 9
- Combinational cloud is replaced by PM

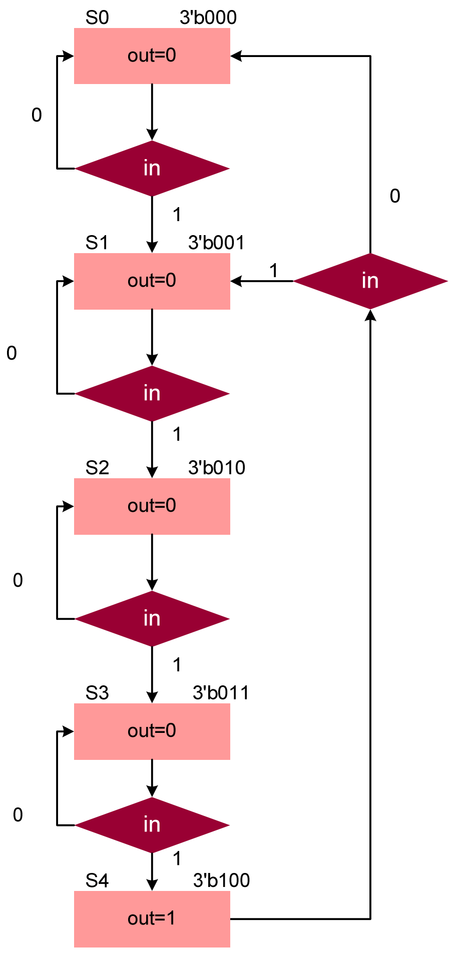


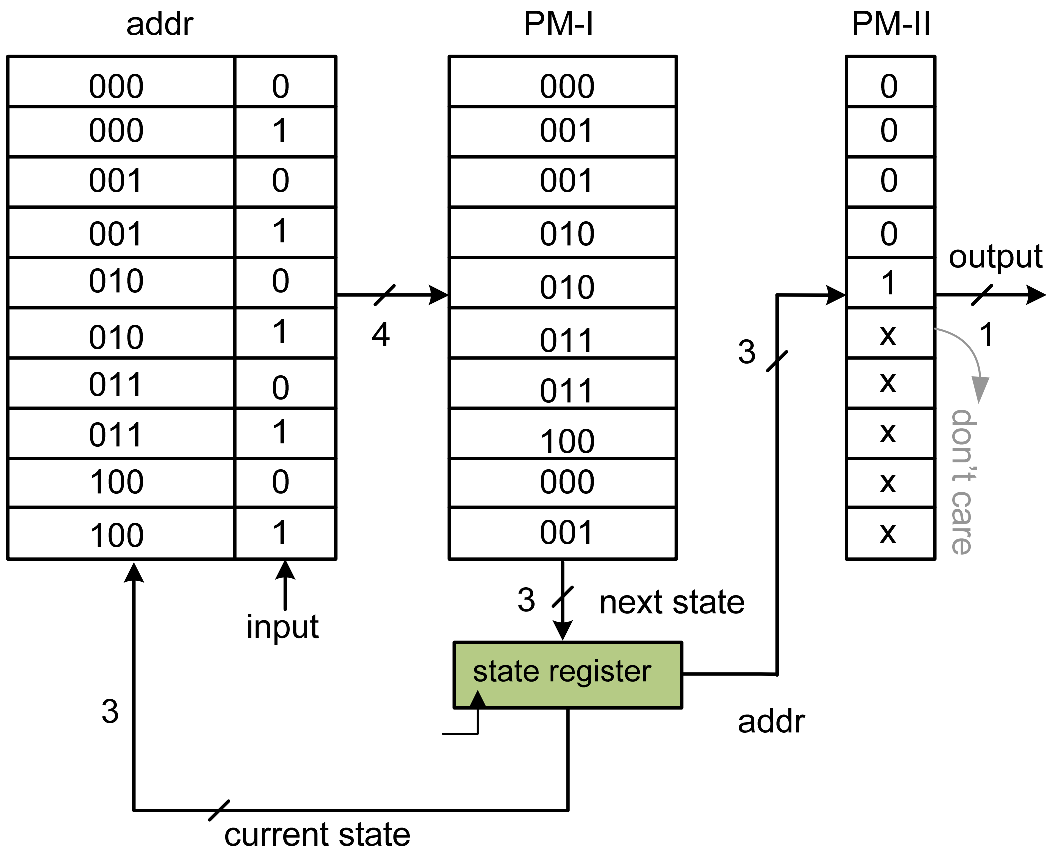


current_state	input	next_state		out
00	0	0	0	0
00	1	0	1	0
01	0	0	1	0
01	1	1	0	0
10	0	1	0	0
10	1	1	1	0
11	0	1	1	0
11	1	0	0	1

Moore Machine

- Number of 1s detected problem of Chapter 9
- Moore machine two combinational clouds are replaced by two PMs





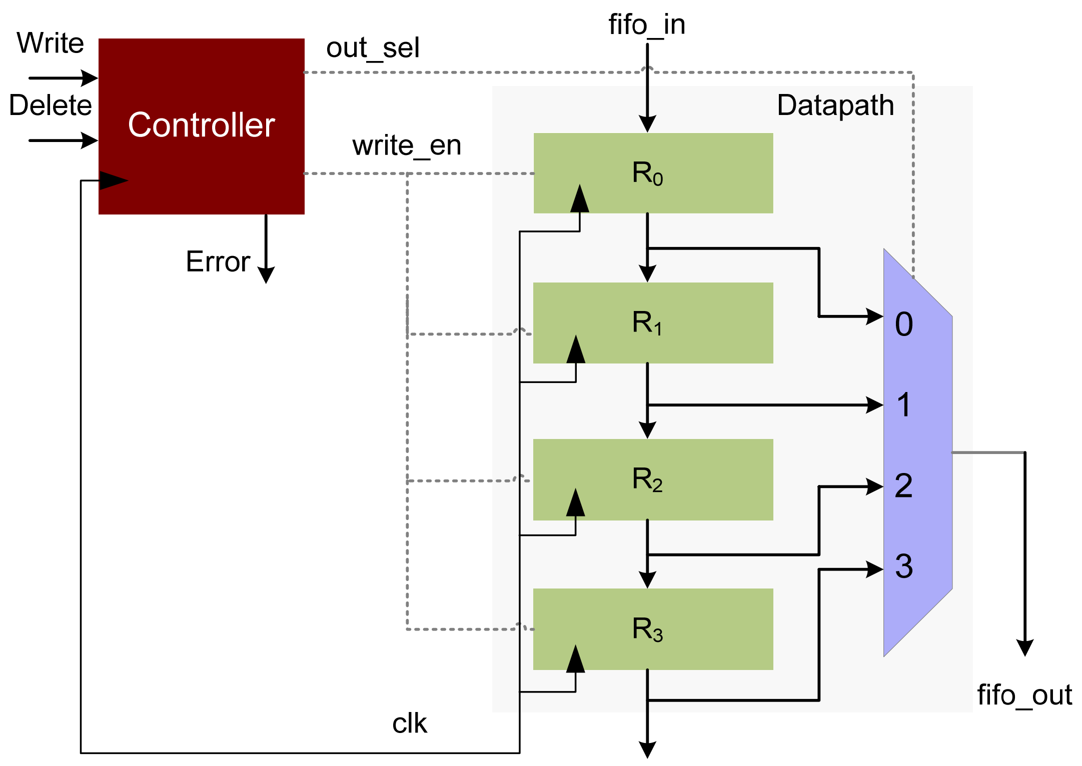
addr		PM-I	PM-II
000	0	000	0
000	1	001	0
001	0	001	0
001	1	010	0
010	0	010	1
010	1	011	x
011	0	011	x
011	1	100	x
100	0	000	x
100	1	001	x

Design Example: 4 Entry FIFO as Micro-program State machine

1. Design a first-in, first-out (FIFO) queue that consists of four registers R0, R1, R2, and R3
2. Write and Delete are the two operations on the queue
3. Write moves data from the fifo_in to R0 that is the tail of the queue
4. Delete deletes the first entry at the head of the queue
5. The head of the queue is available on the fifo_out
6. Writing into a full queue or deletion from an empty queue causes an ERROR condition
7. Assertion of Write and Delete at the same time also causes an ERROR condition

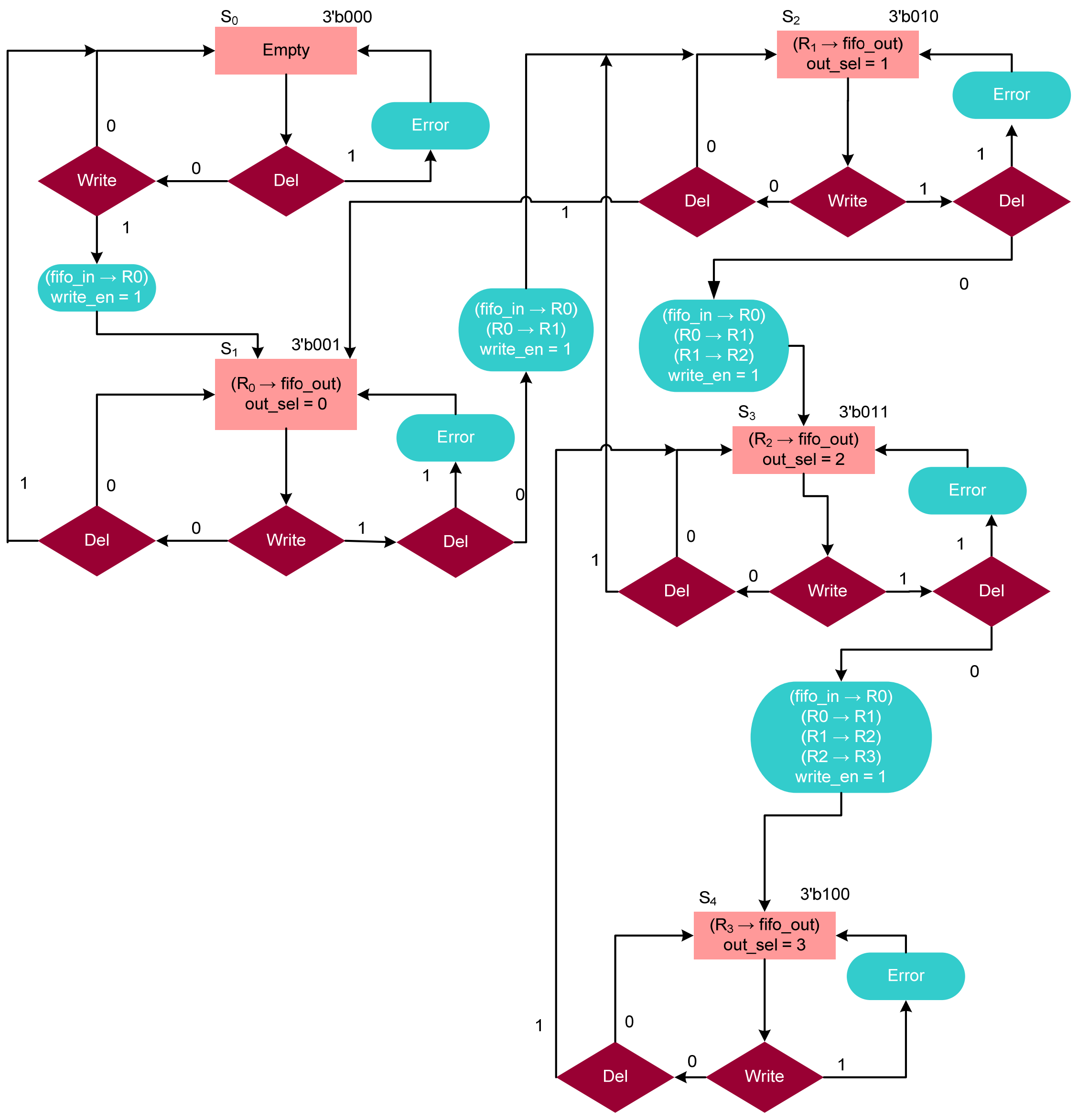
FIFO Design

- Design consists of datapath and controller
- Datapath
  - Four registers
    - Shift registers
  - One 4:1 MUX
- Controller
  - Input signals
    - Write
    - Delete
  - Output signals
    - out_sel
    - write_en
    - Error



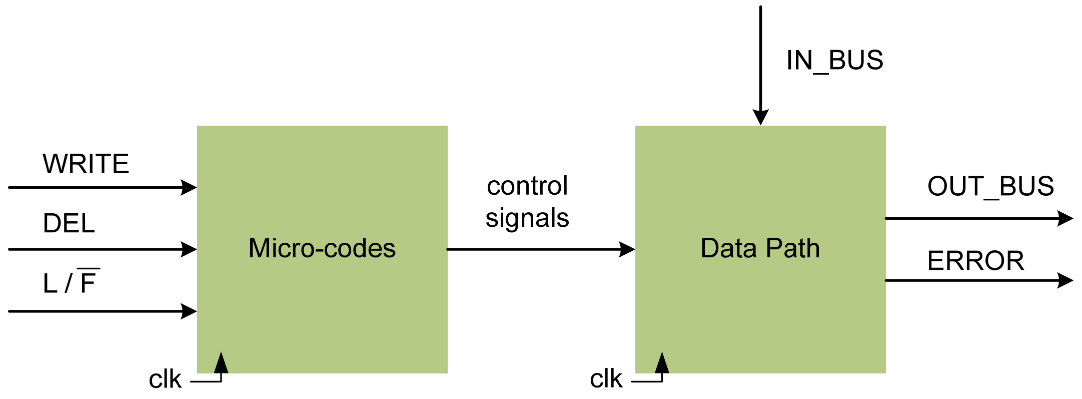
Controller

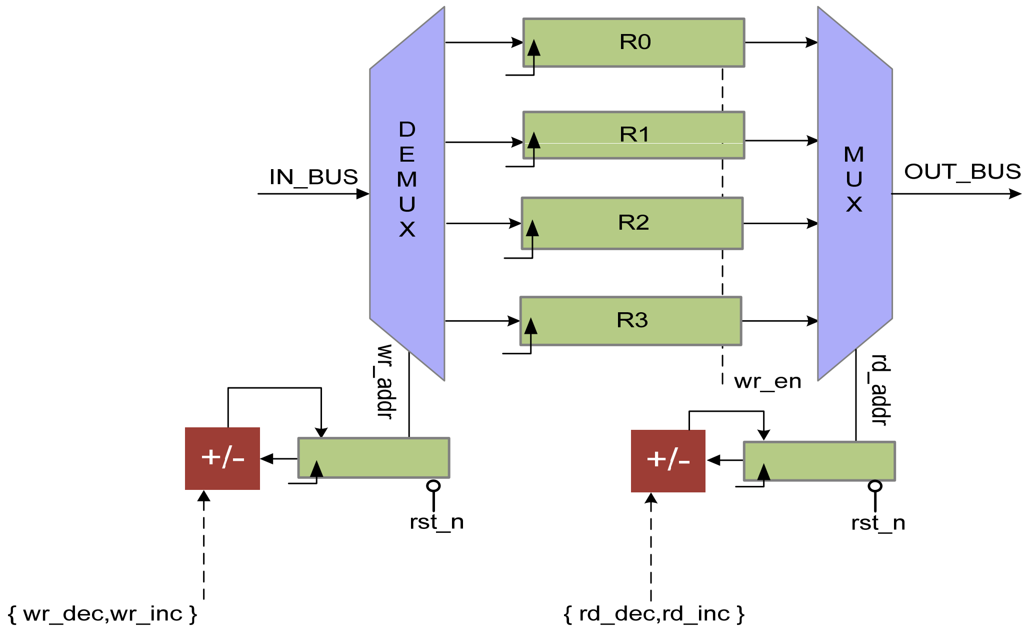
- FSM based design
  - Mealy machine
- Five states
  - S0: Idle
  - S1: One entry
  - S2: Two entries
  - S3: Three entries
  - S4: Full



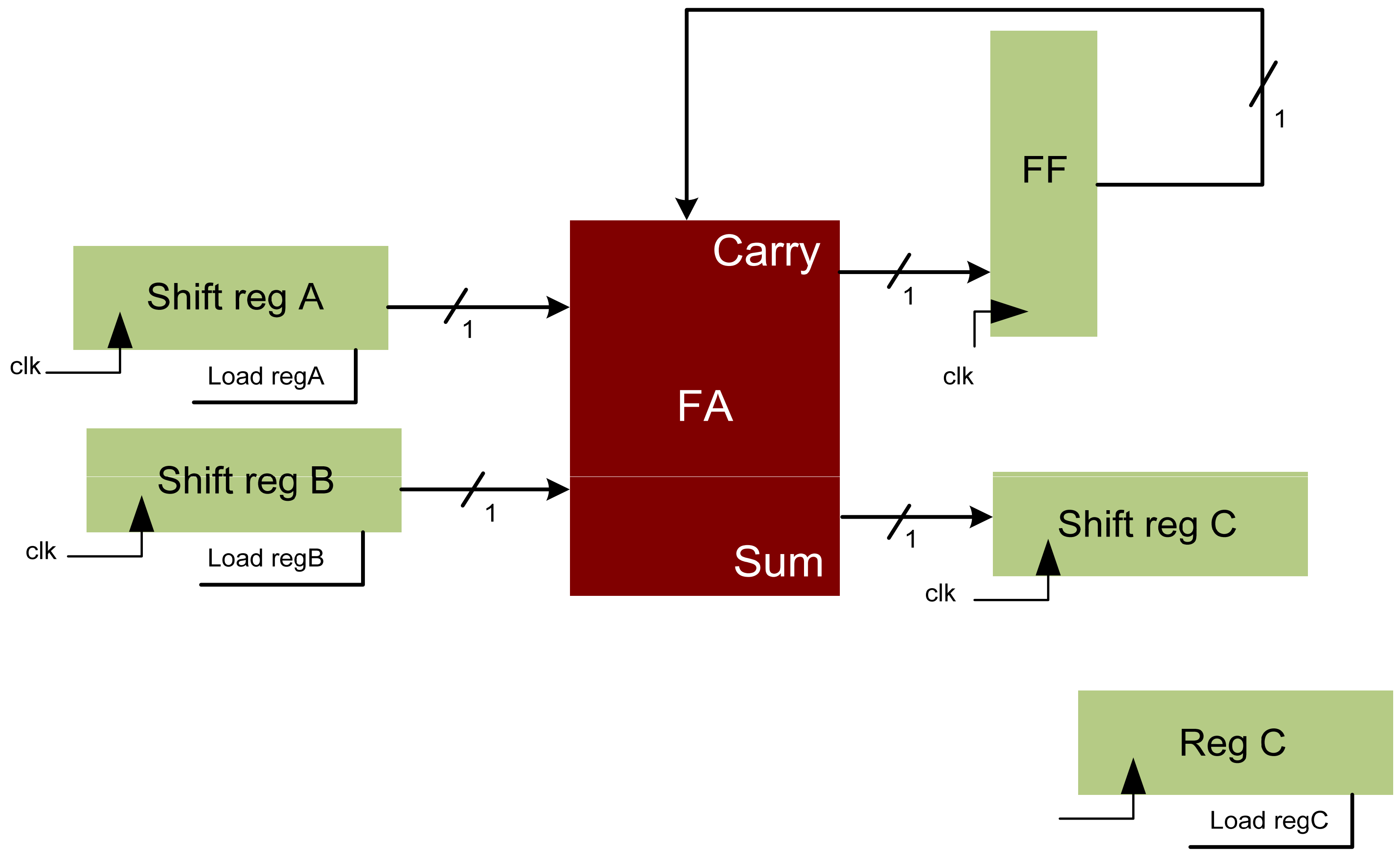
S0 3'b000
Empty
Error
Write
Del
(fifo_in → R0) write_en = 1
S1 3'b001
(R0 → fifo_out) out_sel = 0
S2 3'b010
(R1 → fifo_out) out_sel = 1
(fifo_in → R0) (R0 → R1) write_en = 1
(fifo_in → R0) (R0 → R1) (R1 → R2) write_en = 1
S3 3'b011
(R2 → fifo_out) out_sel = 2
(fifo_in → R0) (R0 → R1) (R1 → R2) (R2 → R3) write_en = 1
S4 3'b100
(R3 → fifo_out) out_sel = 3

FIFO/LIFO

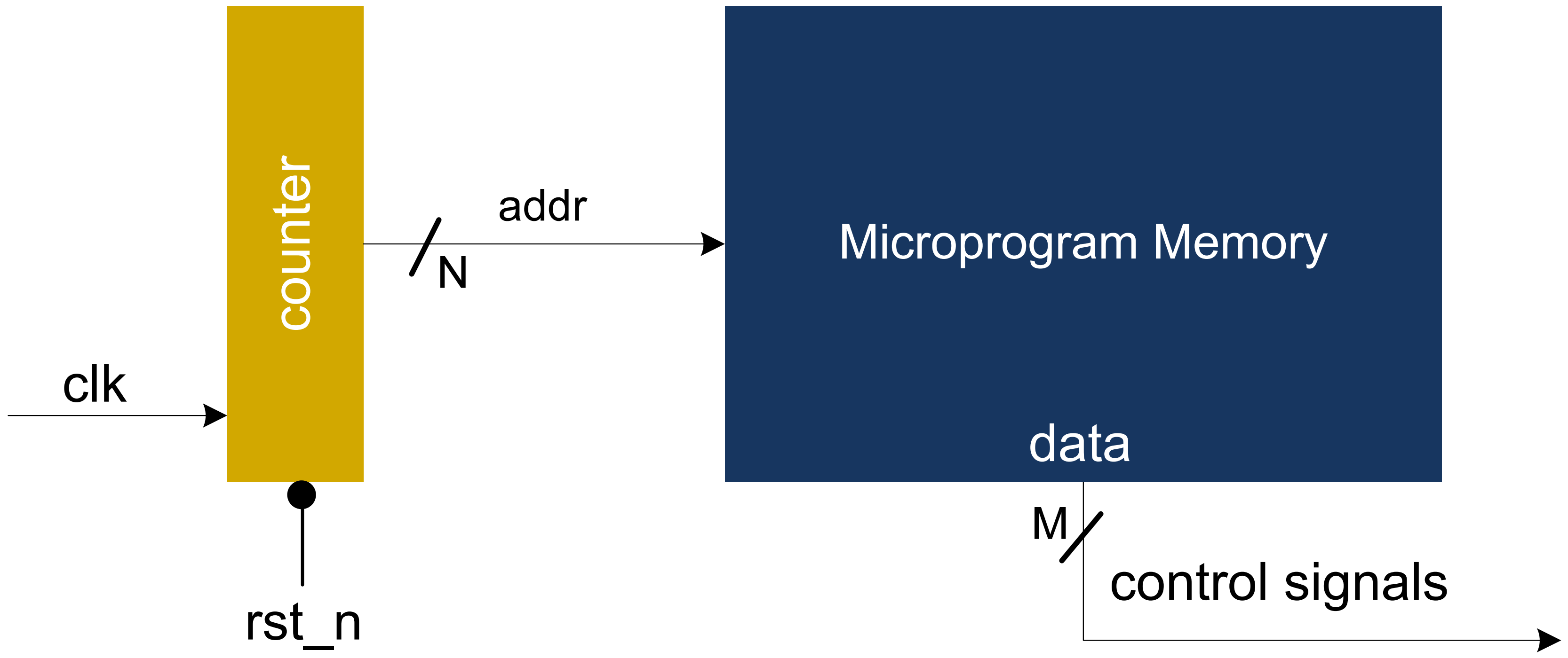

IN_BUS
WRITE
DEL
L / F̄
Micro-codes
control signals
Data Path
OUT_BUS
ERROR
clk
clk


R0
R1
R2
R3
IN_BUS
DEMUX
MUX
OUT_BUS
wr_addr
rd_addr
wr_en
+/-
+/-
rst_n
rst_n
{ wr_dec,wr_inc }
{ rd_dec,rd_inc }

Counter-based FSM: Bit serial adder



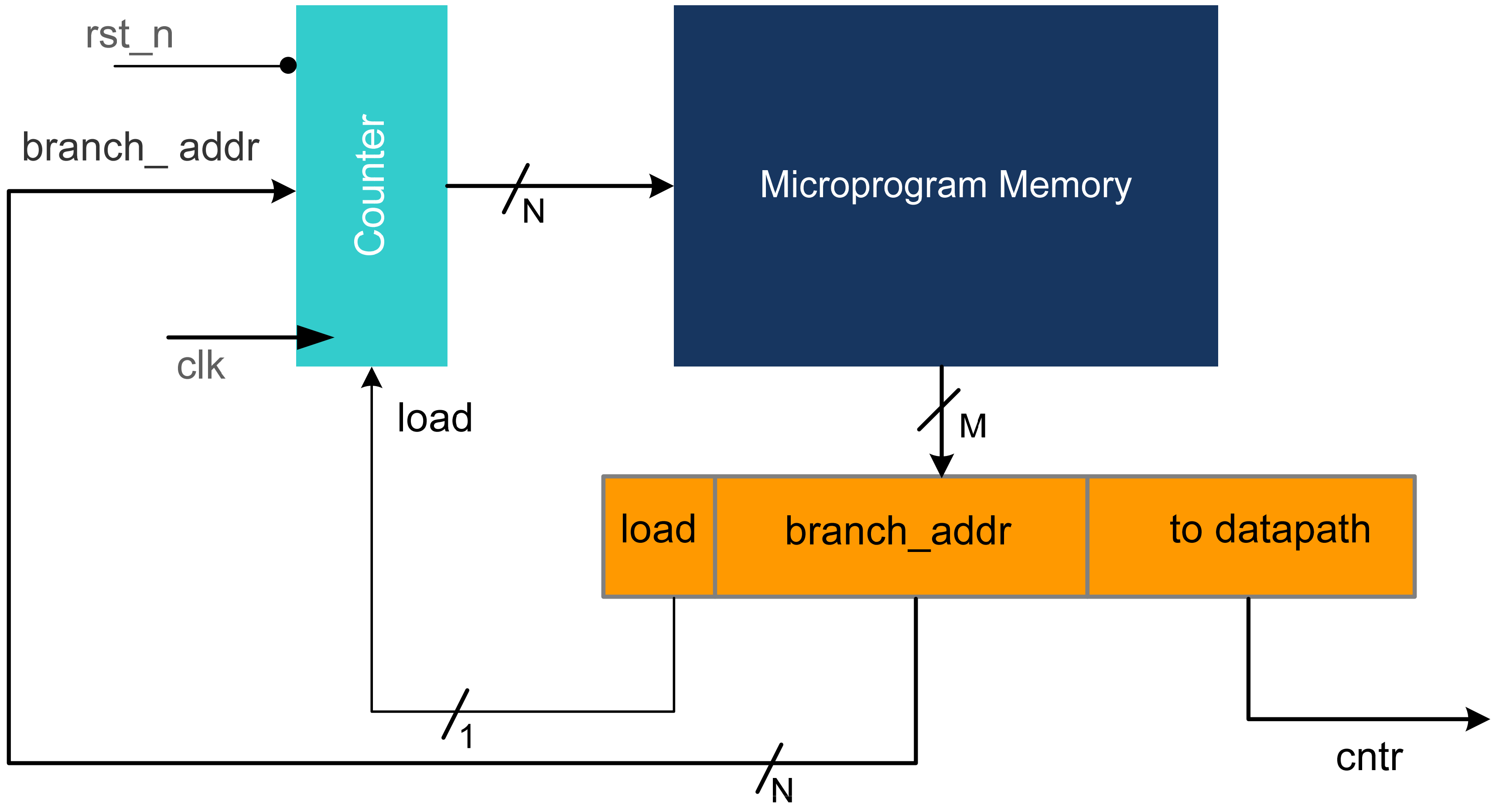
Counter-based State Machine Implementation



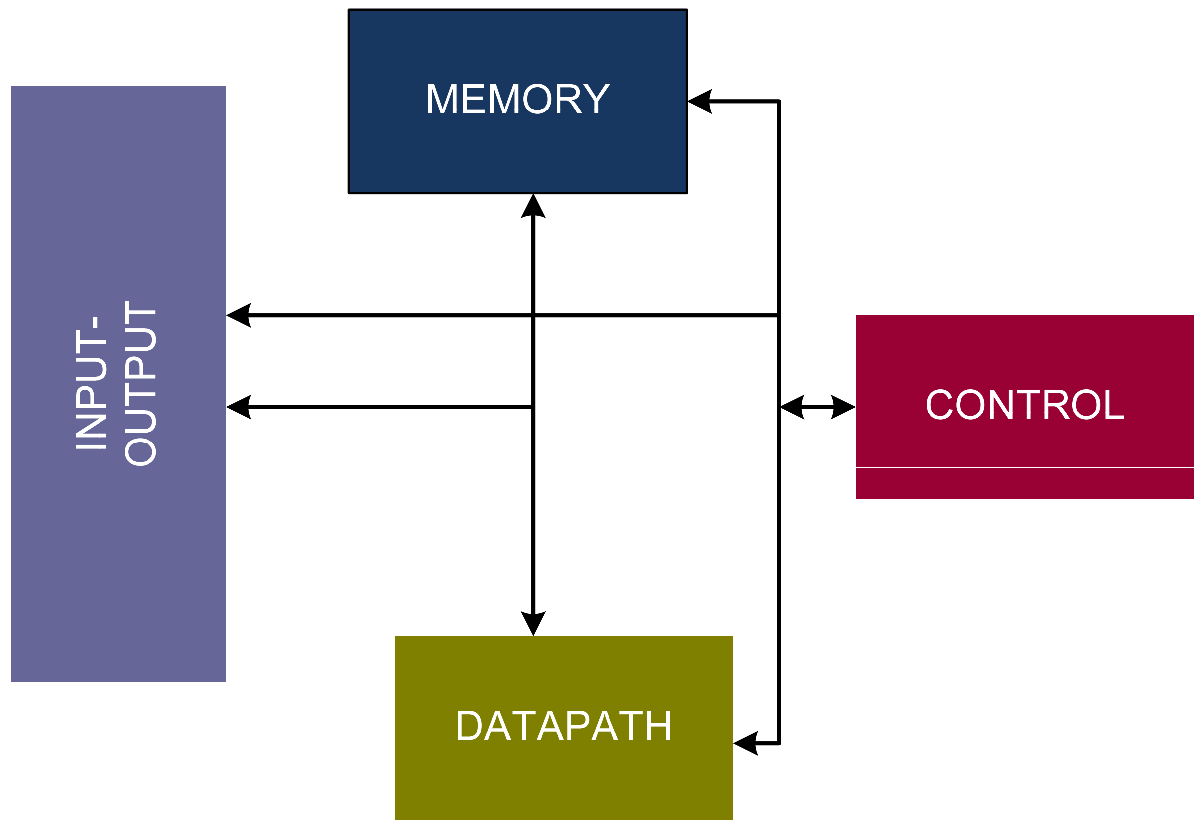
Modifications to Counter-based Microcontrollers 1

- Mechanism for changing count sequence
- Begin another sequence under control of micro-program memory

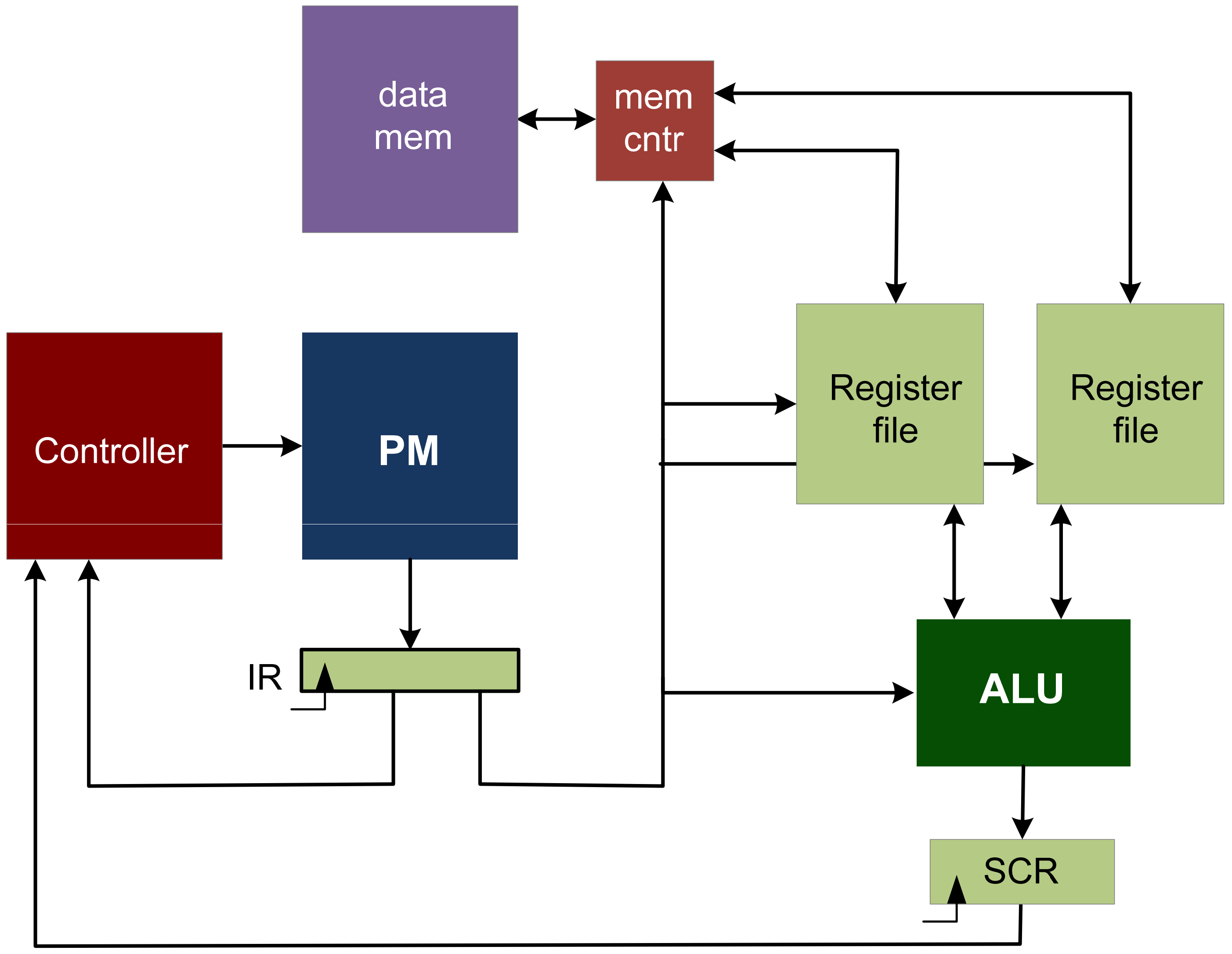
Loadable Counter for *Goto* capability



A Generic Computing Architecture



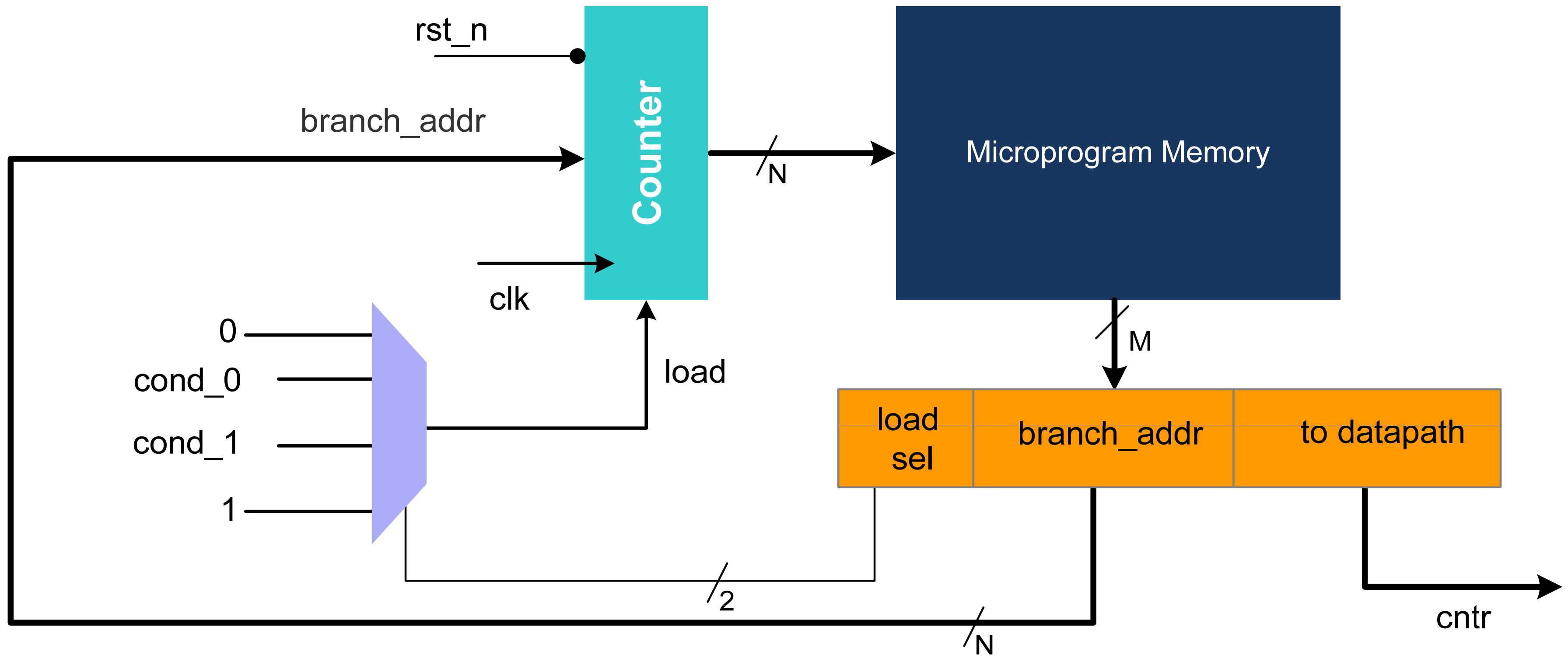
Example: Complete Design



Conditional Execution

- 2-bit control allowing on selection of two conditional inputs and two signals true and false
- The conditions come from the status register in the data path
- Example:
  - zero and positive flags are used for providing conditional execution
  - if (r1 > r2) jump label3; // compute r1-r2 jump if result is positive
  - if (r1 == r2) jump label1; // compute r1-r2, jump if result is zero
  - jump label 2 // unconditional jump

Conditional Branching Capability



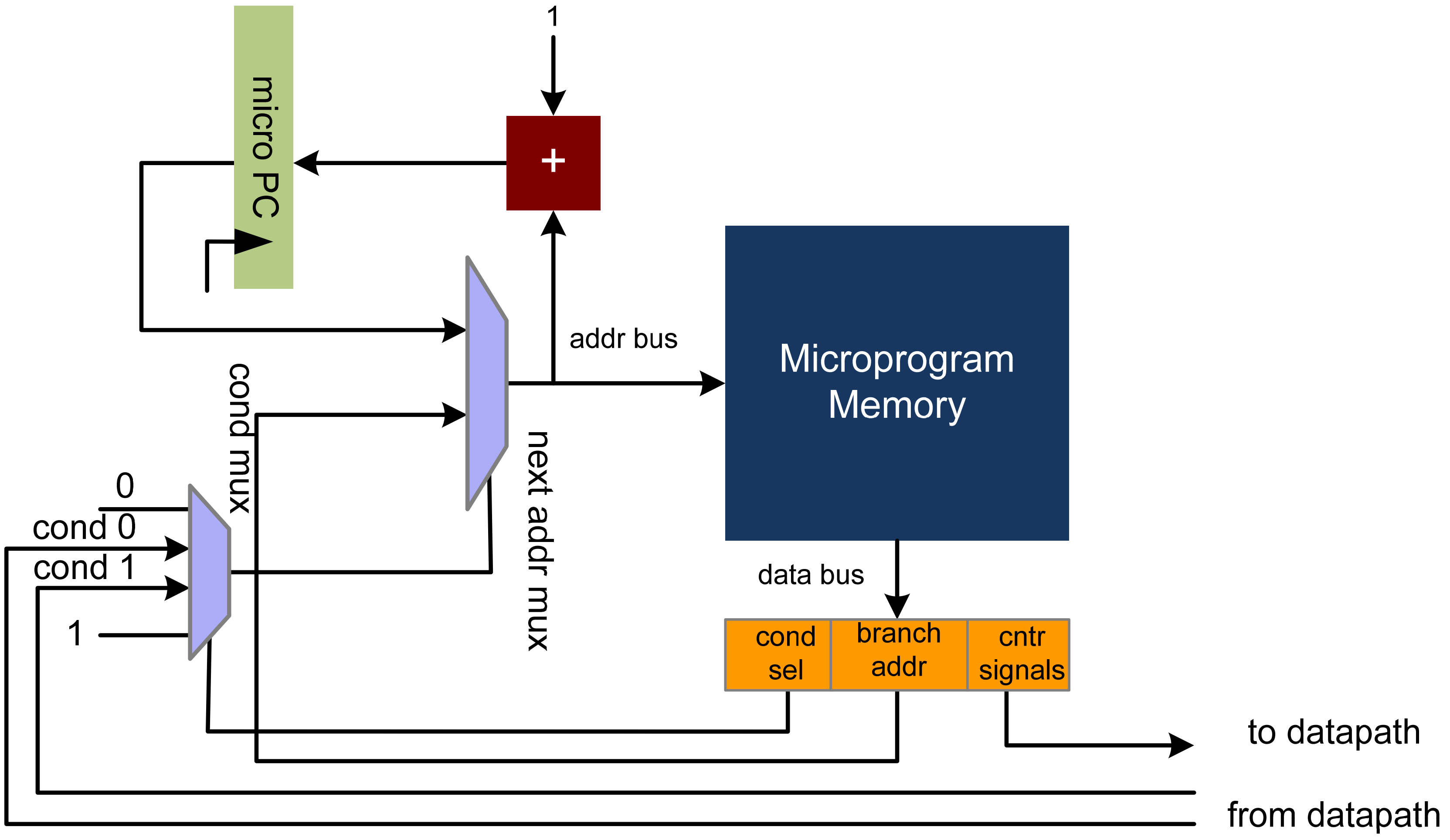
Conditional load

sel	load
00	FALSE (normal execution: never load branch address)
01	COND0 (load branch address if COND0 is TRUE)
10	COND1 (load branch address if COND1 is TRUE)
11	TRUE (unconditional jump: always load branch address)

Pipelined Register

- Often counters are replaced with an ALU based program counter register
- The critical path of the design is long as it goes from the counter to ROM to architecture to functional units generating COND0 and COND1 to conditional MUX.
- The critical path can be broken by inserting a pipeline register in the design

Micro PC Based Controller

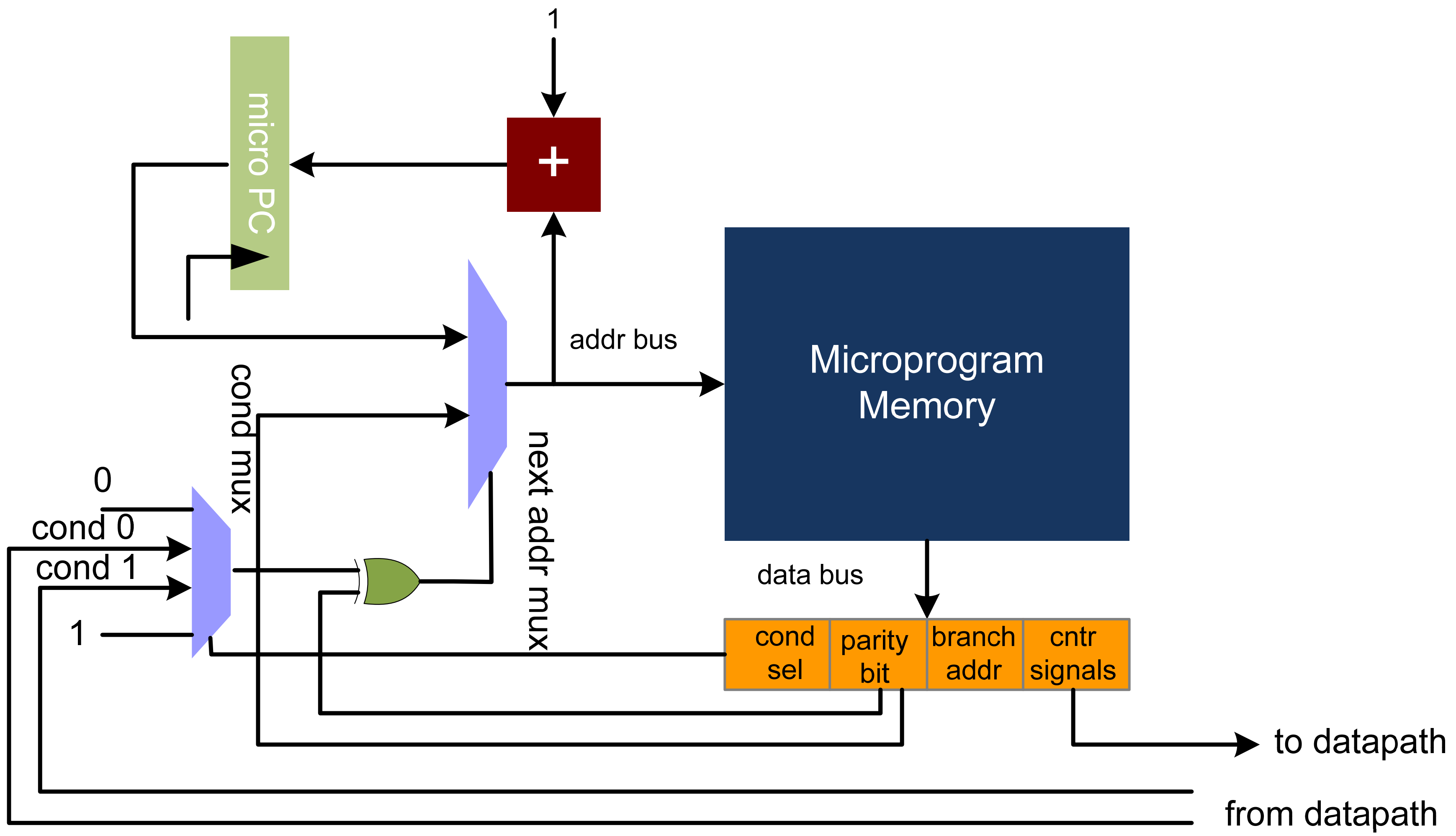


Conditional Branching with Parity

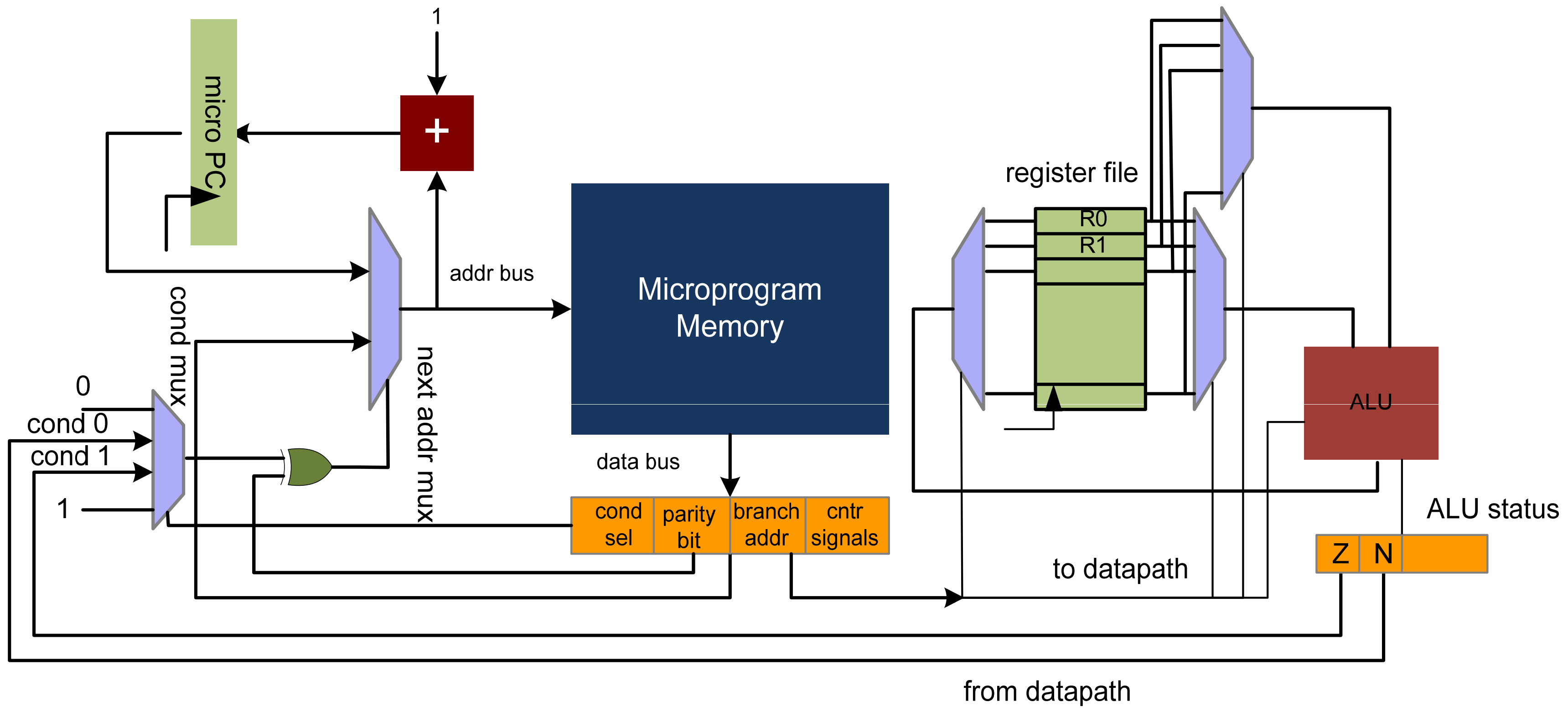
- Conditional execution
  - 2-bit control allowing on selection of two conditional inputs and two signals true and false.
  - A parity bit for inverse selection of the condition

  - Example:
    - zero and positive flags are used for providing conditional execution
    - Inverse selection is provided by parity bit
    - if (r1 > r2) jump label3; // compute r1-r2 jump if result is positive
    - if (r1 < r2) jump label3; // compute r1-r2 jump if result is negative (positive flag 0 and polarity bit is set)
    - if (r1 == r2) jump label1; // compute r1-r2, jump if result is zero
    - if (r1 != r2) jump label1; // compute r1-r2, jump if result is not zero (zero flag is 0 and polarity bit is set)
    - jump label 2 // unconditional jump

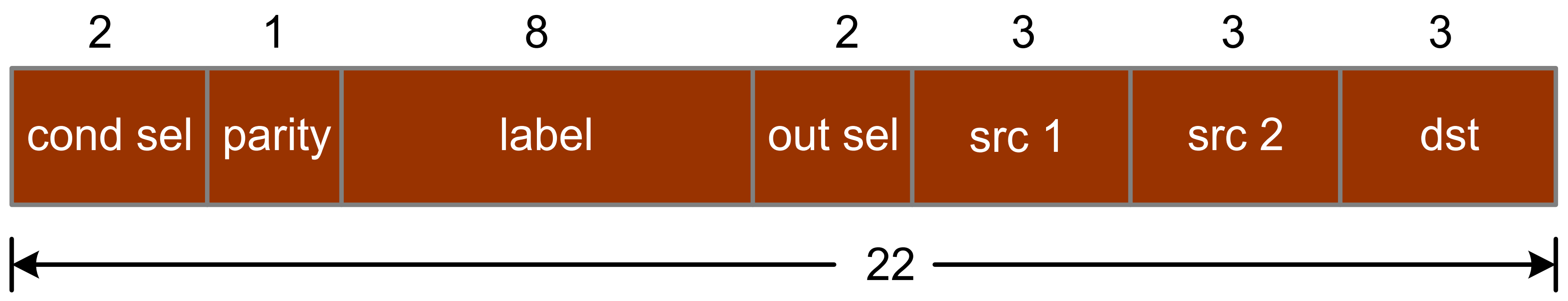
Addition of Parity bit

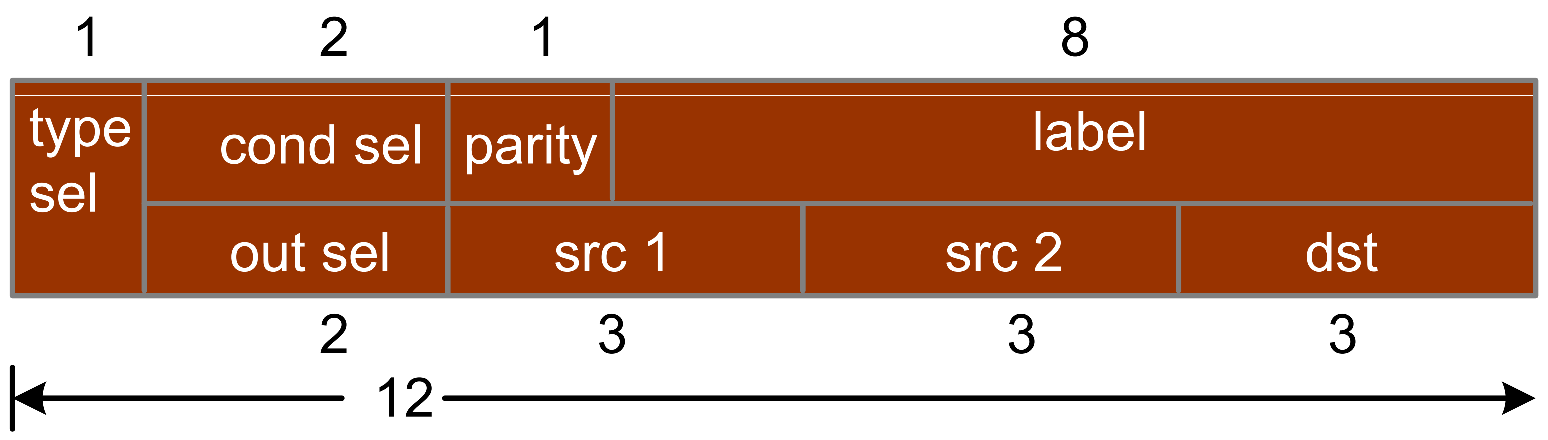


Controller with data path

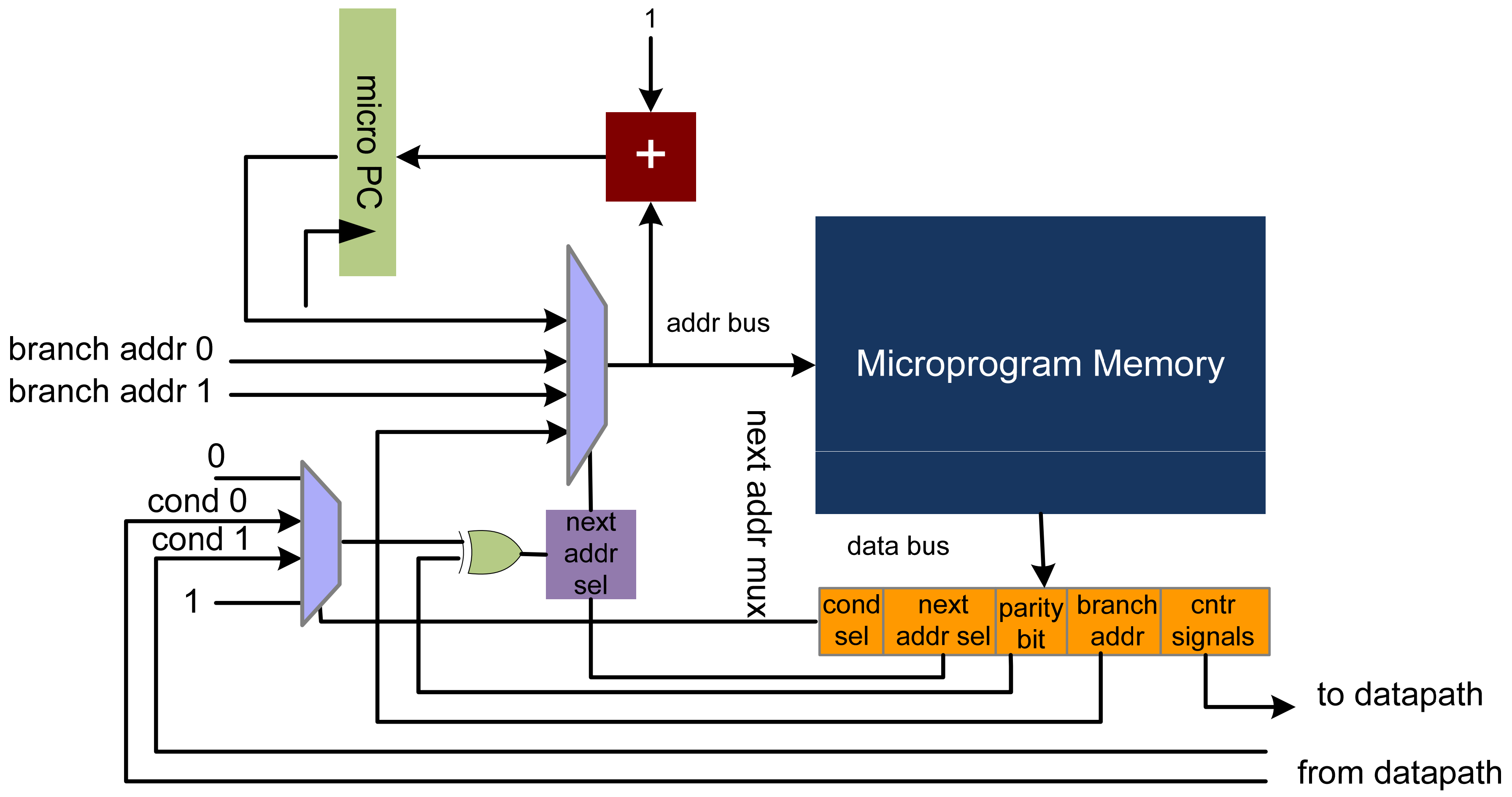


Instruction word design





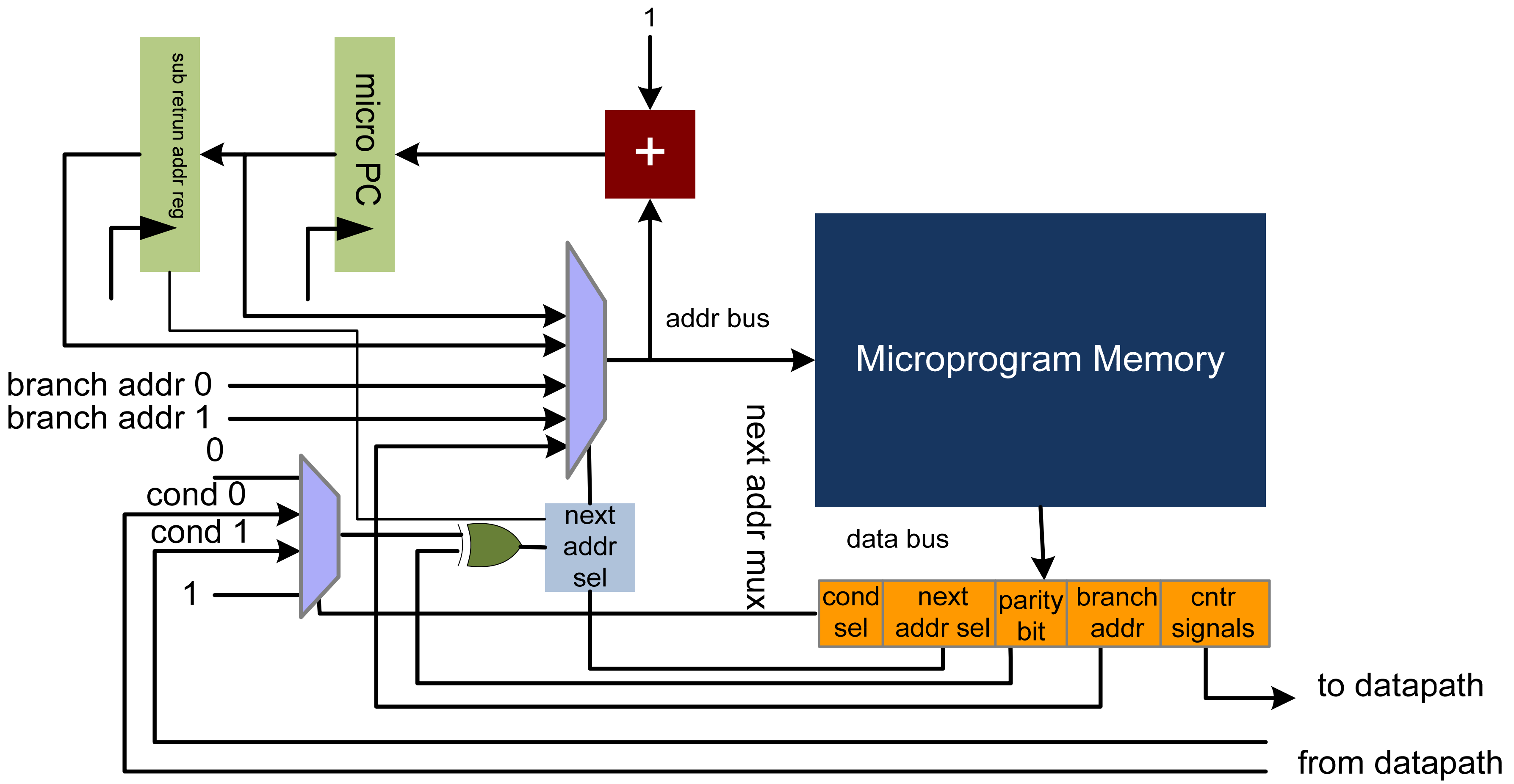
micro PC-based ASM with Conditional Multiplexer



Subroutine Execution

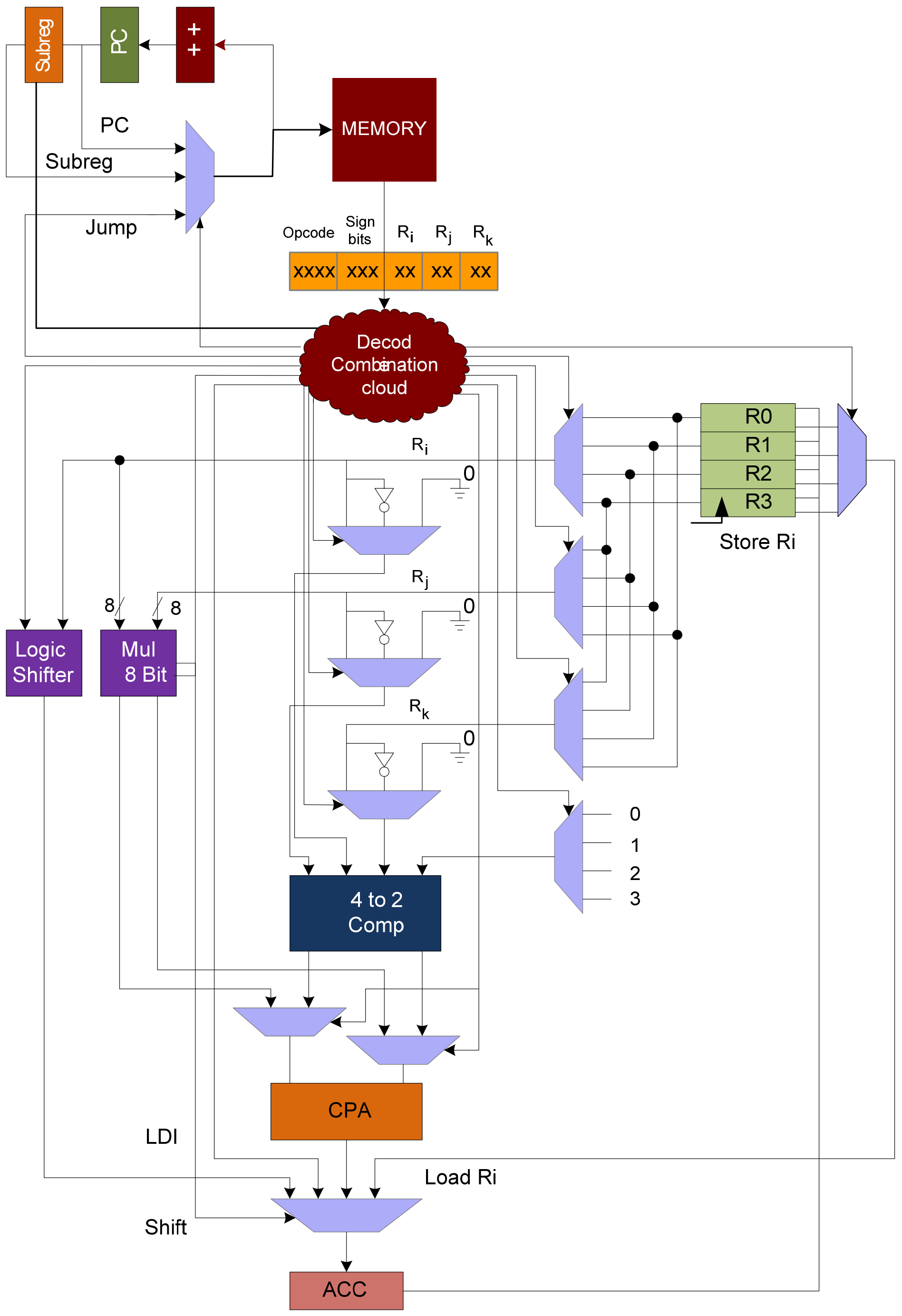
- Temporary storage location added for the copy of contents of mPC register
  - Register referred to as Subroutine Return Address Register
- Address kept in register allows on returning to next address of microprogram execution after complete subroutine call

Register-based Controller with Subroutine Capability

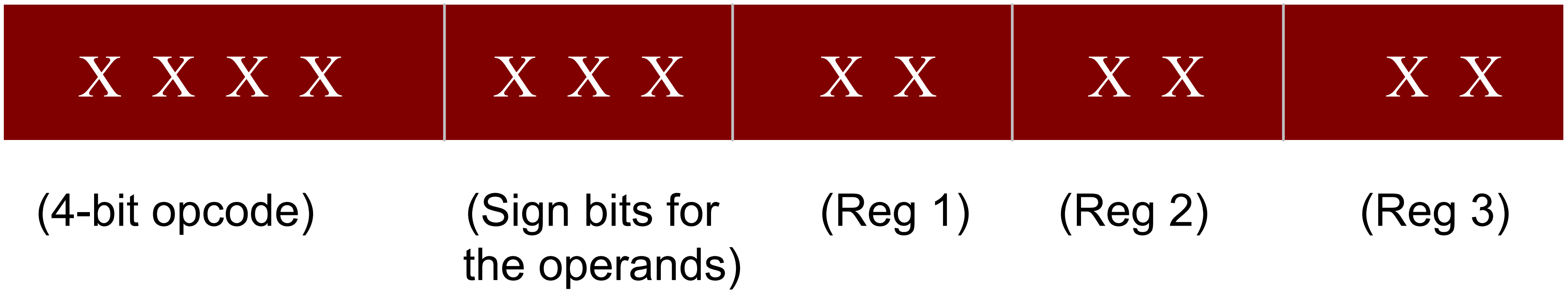


Design Problem

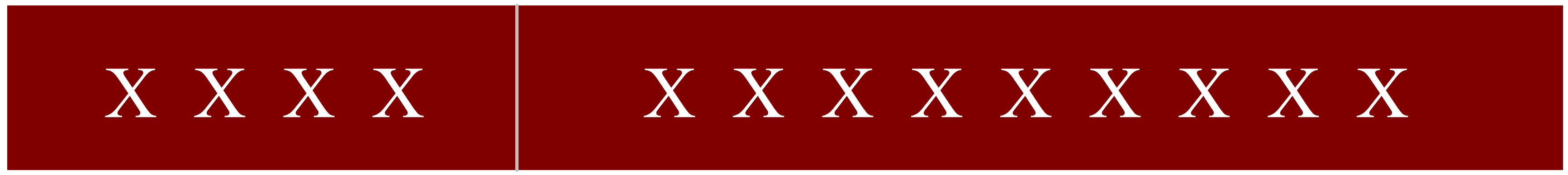
Opcode	Instruction	Description
0000	$ACC = R_i + c$	Add a constant into R_i
0001	$ACC = \pm R_i \pm R_j$	Signed addition of two registers
0010	$ACC = \pm R_i \pm R_j \pm R_k$	Signed addition of three registers
0011	$ACC = R_i + R_j + R_k + c$	Addition of three registers and a constant
0100	$ACC = R_i * R_j$	Multiplication of 8 LSBs of two registers
0101	Load R_i	Load register into accumulator
0110	Store R_i	Store the contents of the accumulator to register
0111	Load n	Load an integer into accumulator
1000	Branch if Acc = 0	Branch if accumulator is zero
1001	Branch if Acc < 0	Branch if accumulator is negative
1010	Always Branch	Unconditional branch
1011	Call Subroutine	Jump to subroutine address
1100	Return	Return from a subroutine call
1101	$Acc = R_i <<>> n$	Right or left shift a register by 16, result in Accumulator


Subreg
PC
++
PC
Subreg
Jump
MEMORY
Opcode
Sign bits
R_i
R_j
R_k
xxxx
xxx
xx
xx
xx
Decod Combination cloud
R0
R1
R2
R3
Store Ri
R_i
0
R_j
0
R_k
0
8
8
Logic Shifter
Mul 8 Bit
0
1
2
3
4 to 2 Comp
CPA
LDI
Load Ri
Shift
ACC

Arithmetic Instructions



Branch/Load Instruction

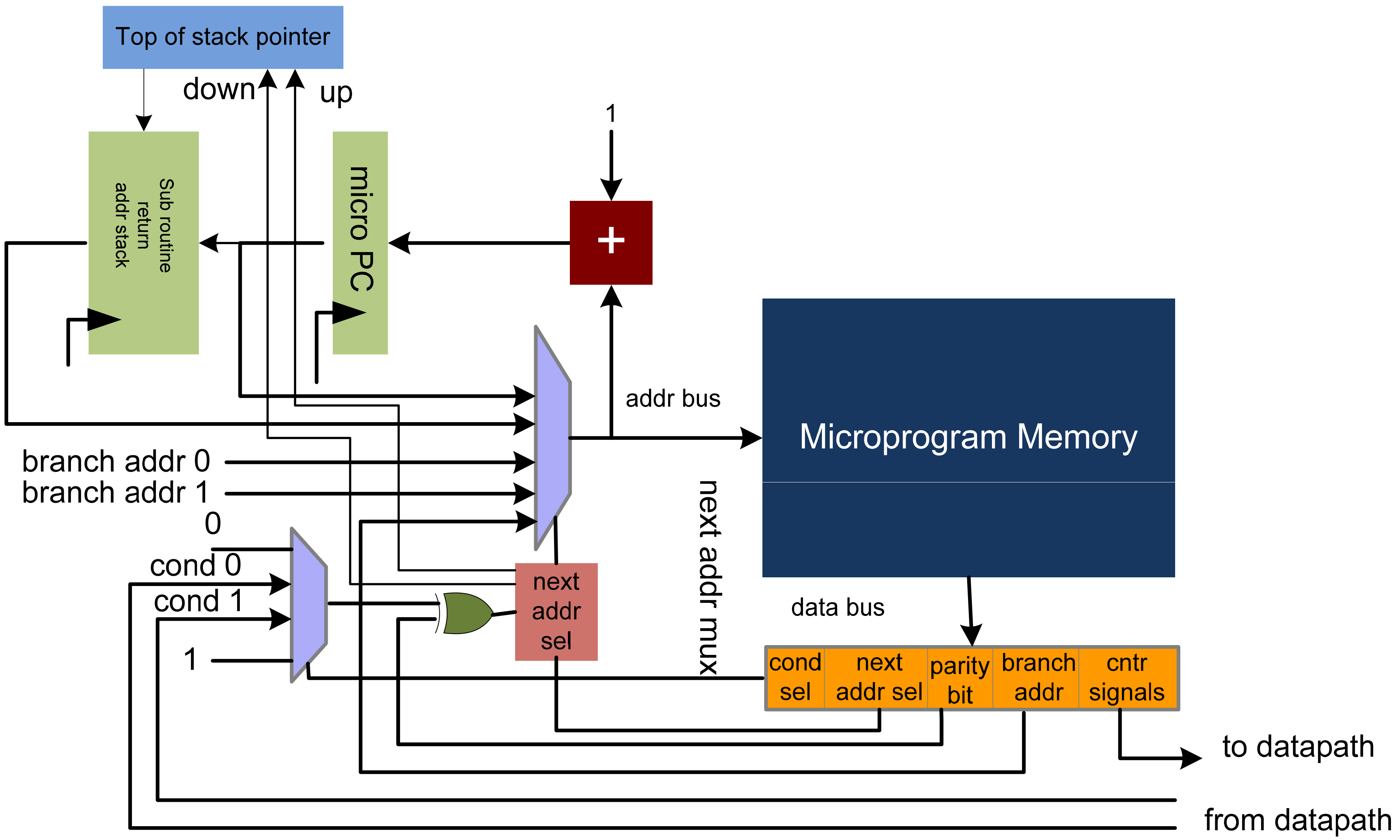


(4-bit opcode) (9-bit Address/Integer)

Nested Subroutine Nested Subroutine Execution

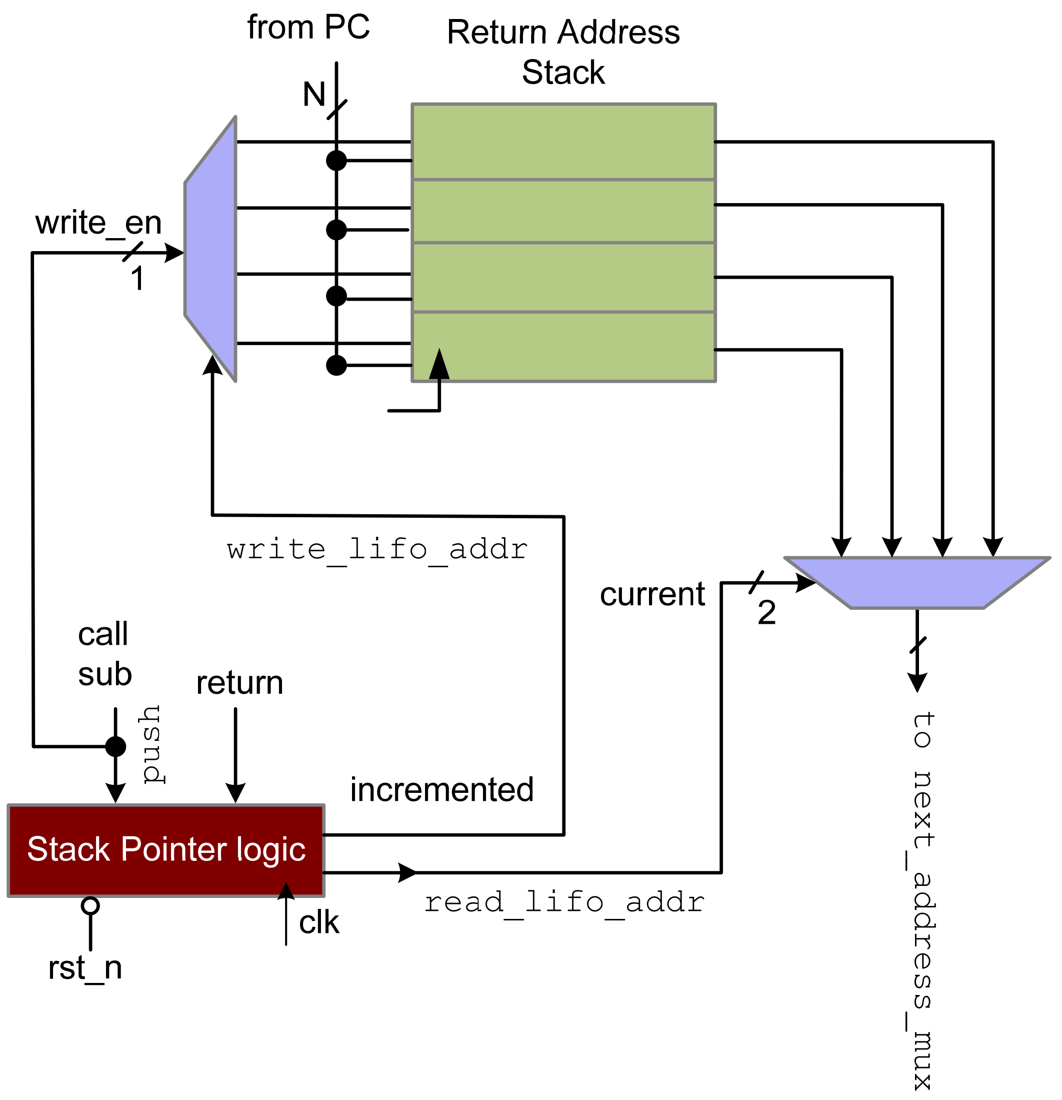
- Subroutine Return Address Register replaced by Subroutine Return Address Memory of stack structure
  - Possibility to store more than one return address – necessity in implementing nested subroutine calls
- Up/down counter (TOP OF STACK) added for stack pointer manipulations

Micro-PC Controller with Nested Subroutine Capability

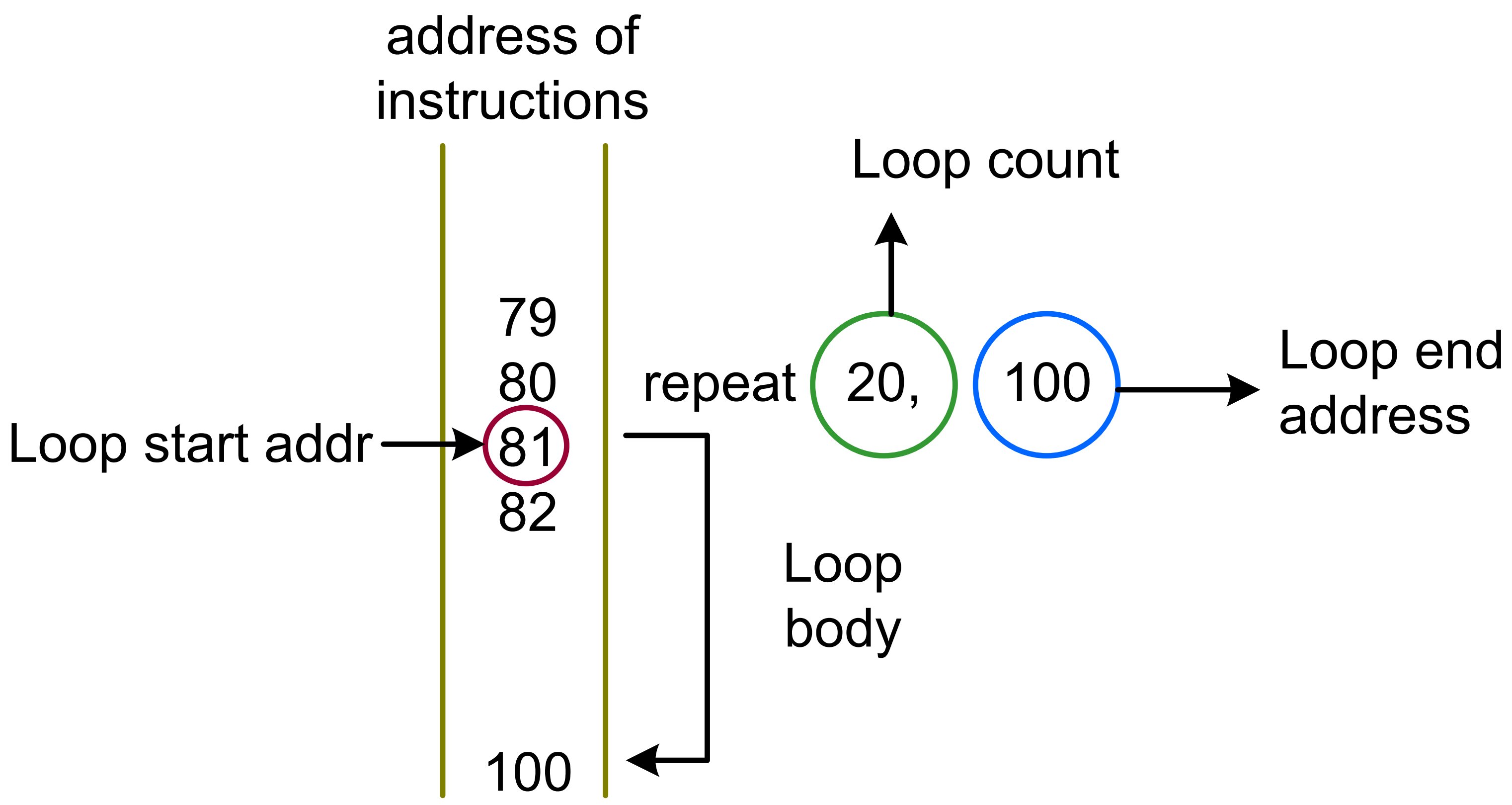


Nested Subroutine Support

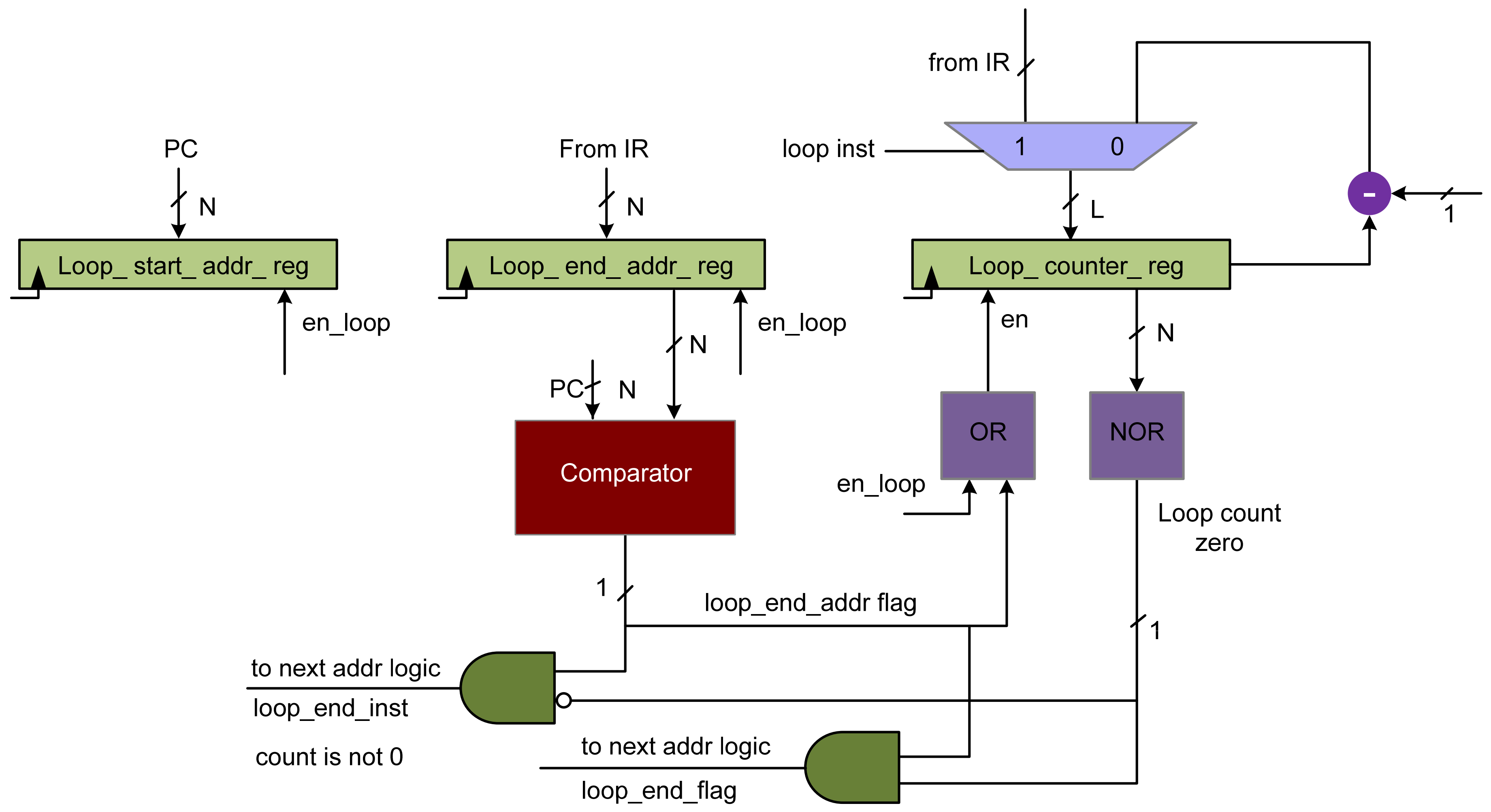
- 4 levels of nesting
- 1 stack
- Stack Pointer Reg.



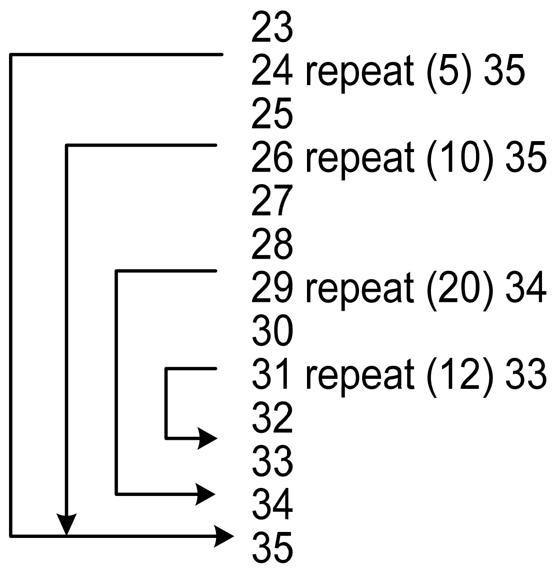
Loop Support



Single Loop Instruction Support

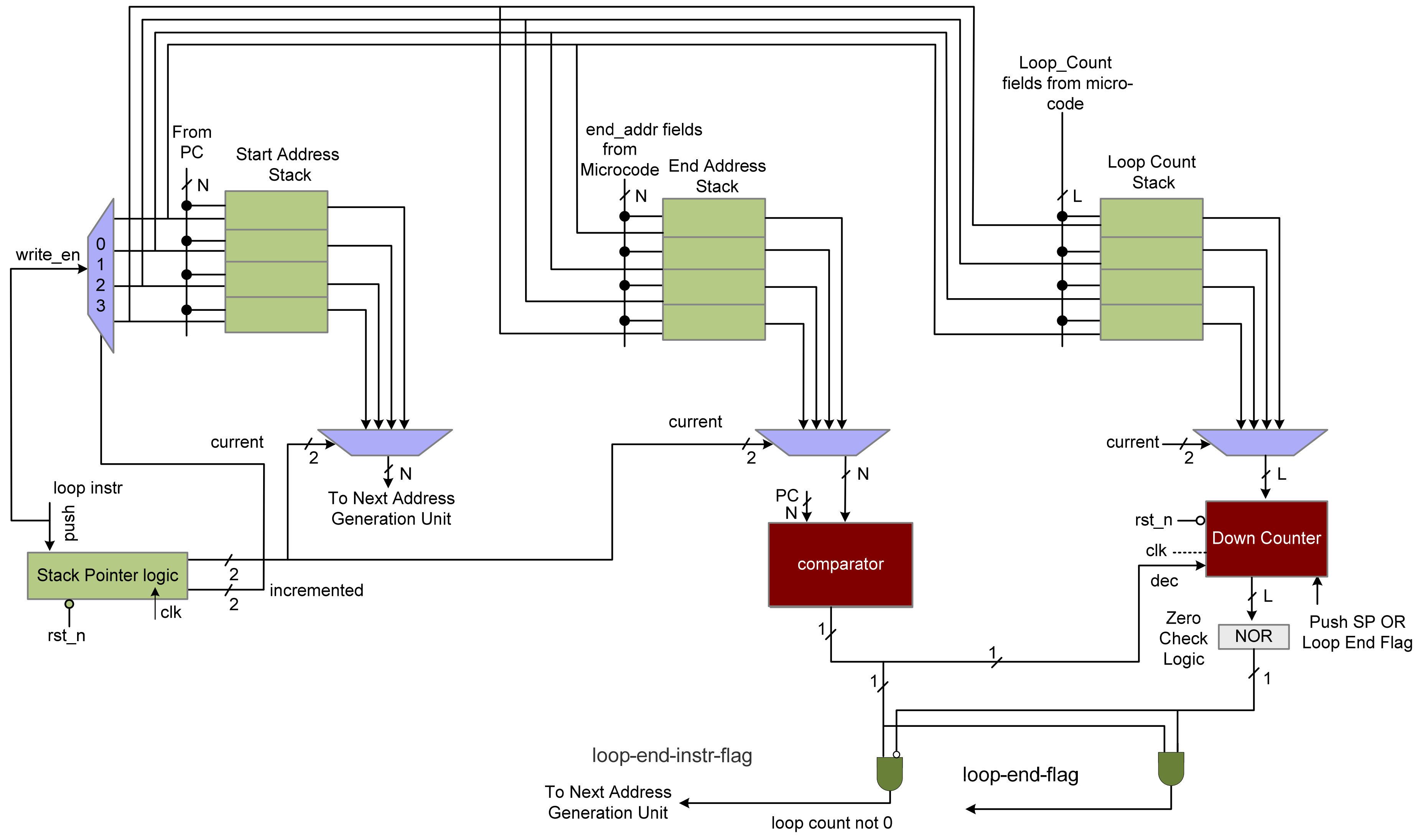


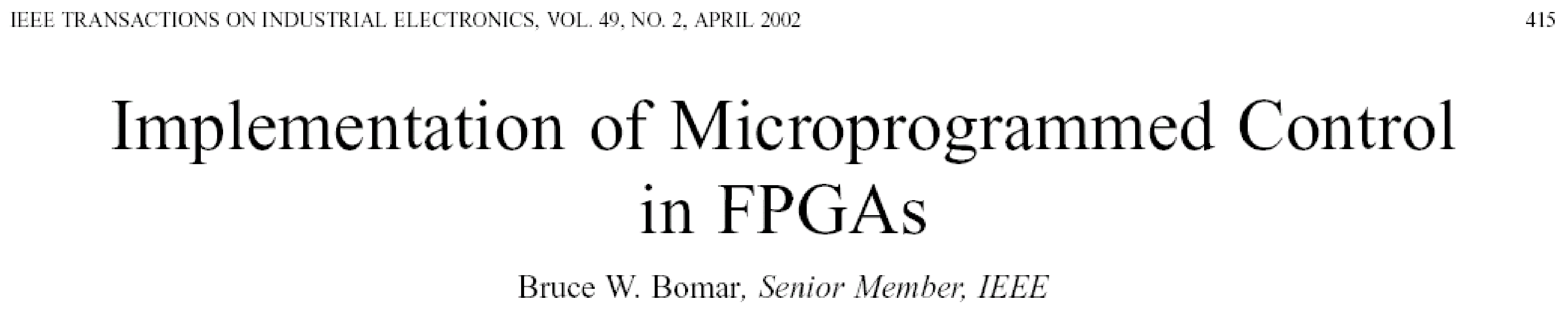
Nested Loop Support



Loop start addr	Loop end addr	Loop count
25	35	5
27	35	10
30	34	20
32	33	12

Loop Machine

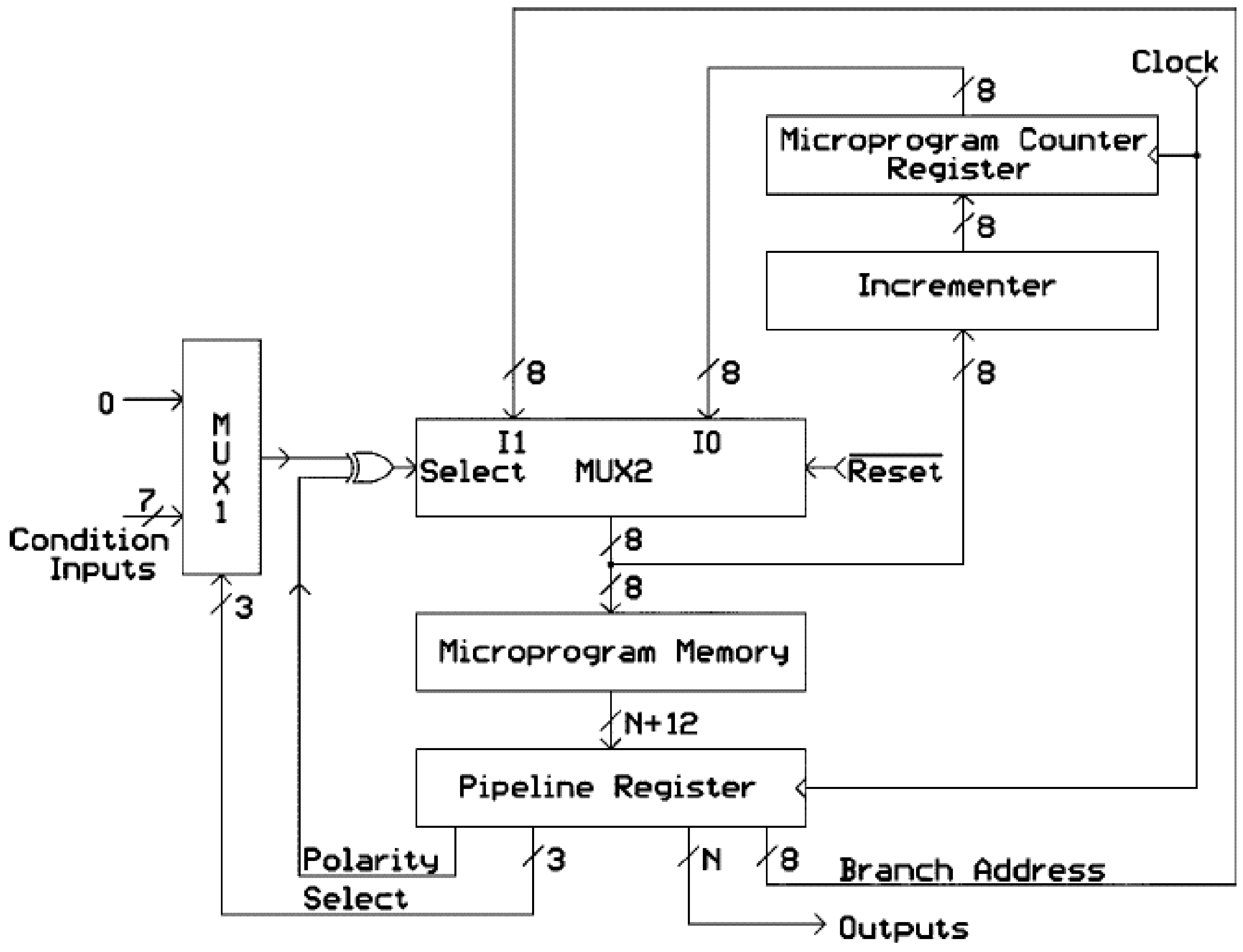


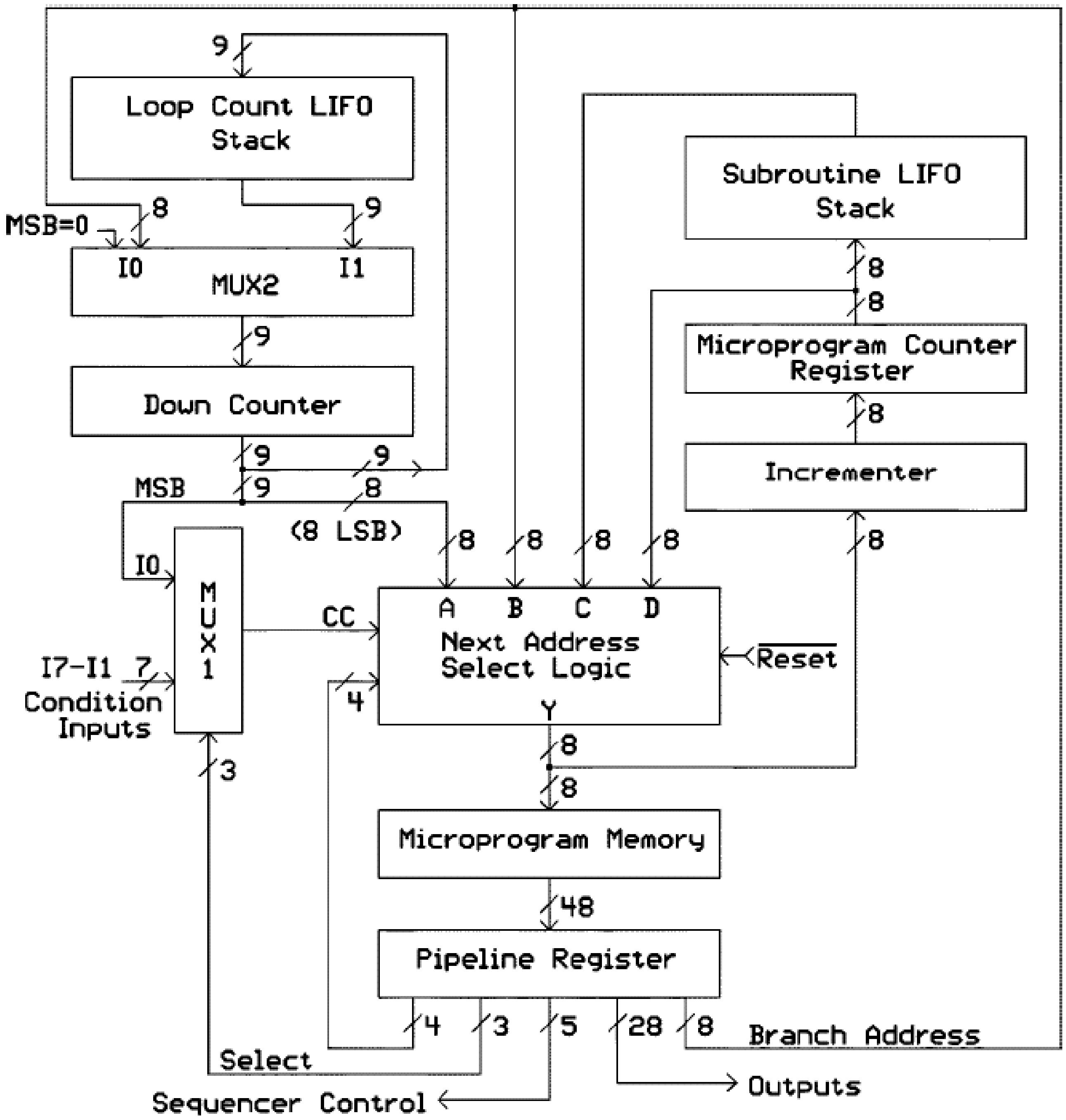


IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, VOL. 49, NO. 2, APRIL 2002 415

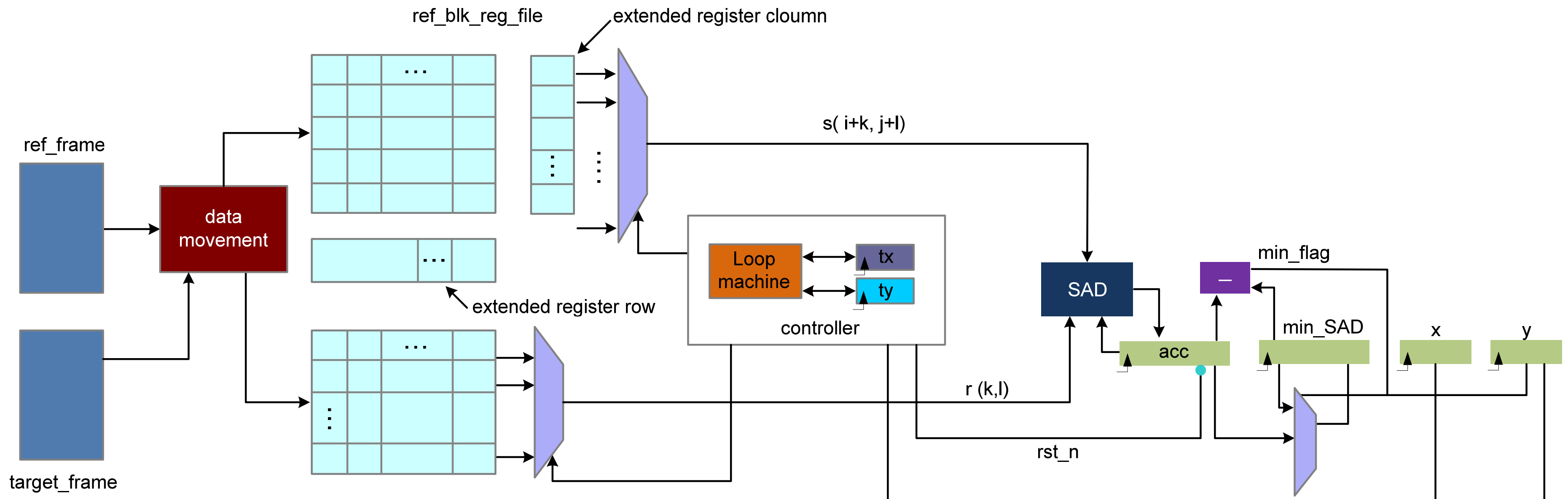
Implementation of Microprogrammed Control in FPGAs

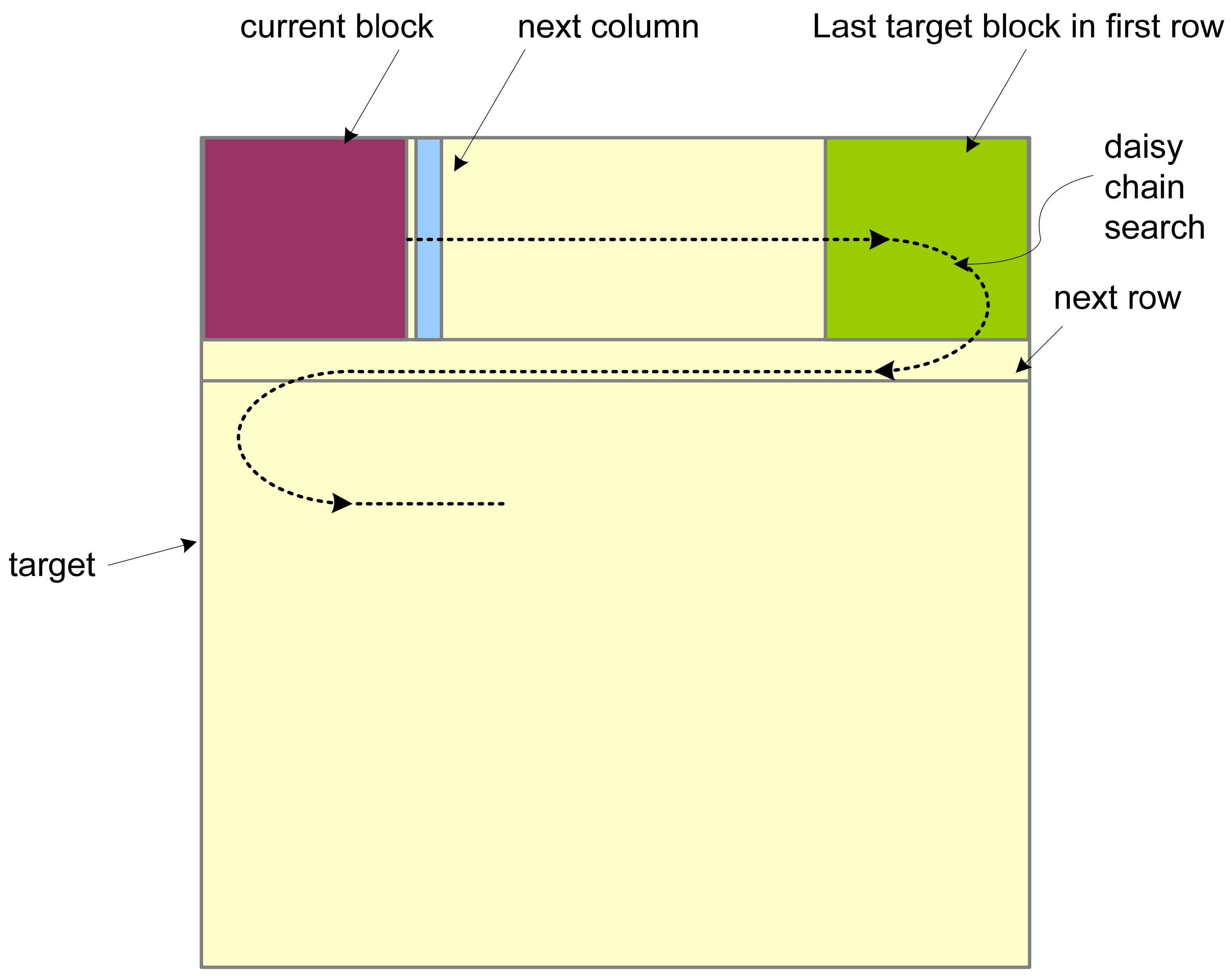
Bruce W. Bomar, *Senior Member, IEEE*


Clock
8
Microprogram Counter Register
8
Incrementer
8
8
8
0
MUX1
7
Condition Inputs
3
I1
I0
Select
MUX2
Reset
8
8
Microprogram Memory
N+12
Pipeline Register
Polarity
Select
3
N
8
Branch Address
Outputs

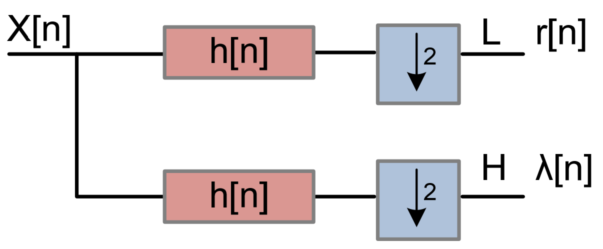

Loop Count LIFO Stack
MSB=0
I0
I1
MUX2
Down Counter
MSB
(8 LSB)
MUX1
CC
I7-I1 Condition Inputs
Select
Subroutine LIFO Stack
Microprogram Counter Register
Incrementer
A B C D
Next Address Select Logic
Y
Reset
Microprogram Memory
Pipeline Register
Branch Address
Outputs
Sequencer Control
9
8
9
9
9
9
9
8
8
8
8
8
8
8
8
8
8
8
7
3
4
48
4
3
5
28
8

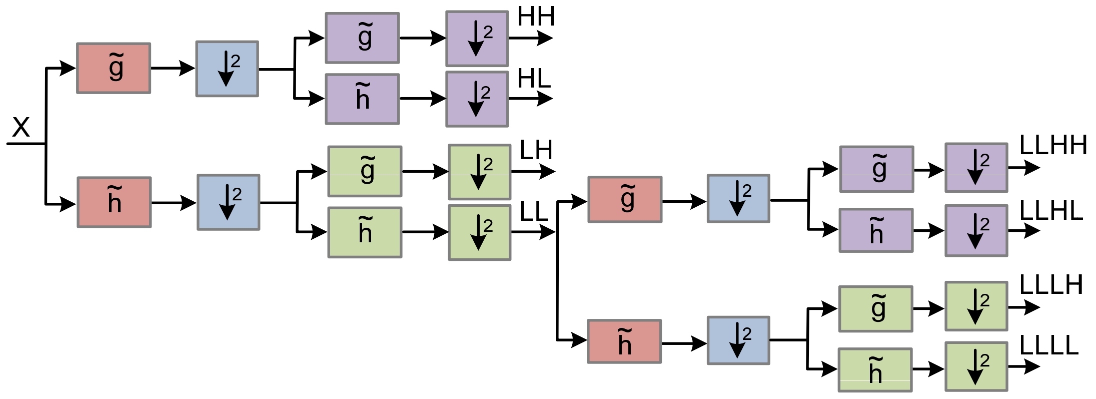
Example: Motion Estimation




current block
next column
Last target block in first row
daisy chain search
next row
target

Example: Design of a Wavelet Processor

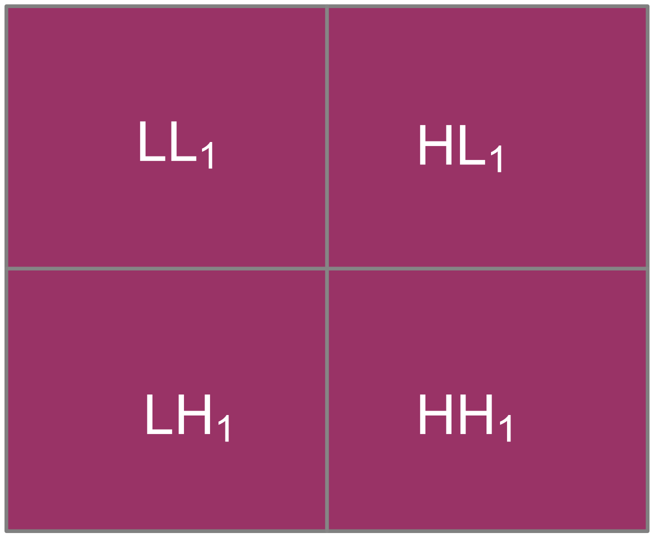




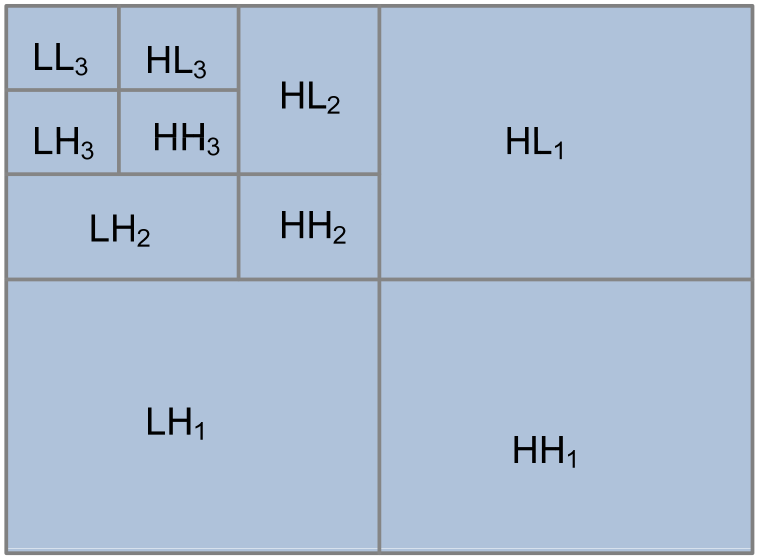
(a) 1-D wavelet decomposition

(b) 2-level 2-D pyramid decomposition

Example: Design of a Wavelet Processor



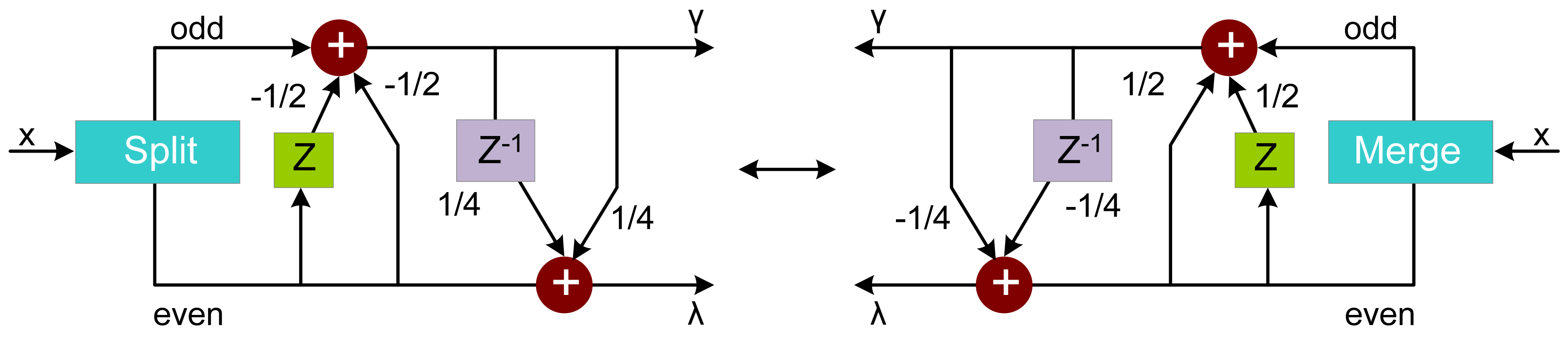
(a)



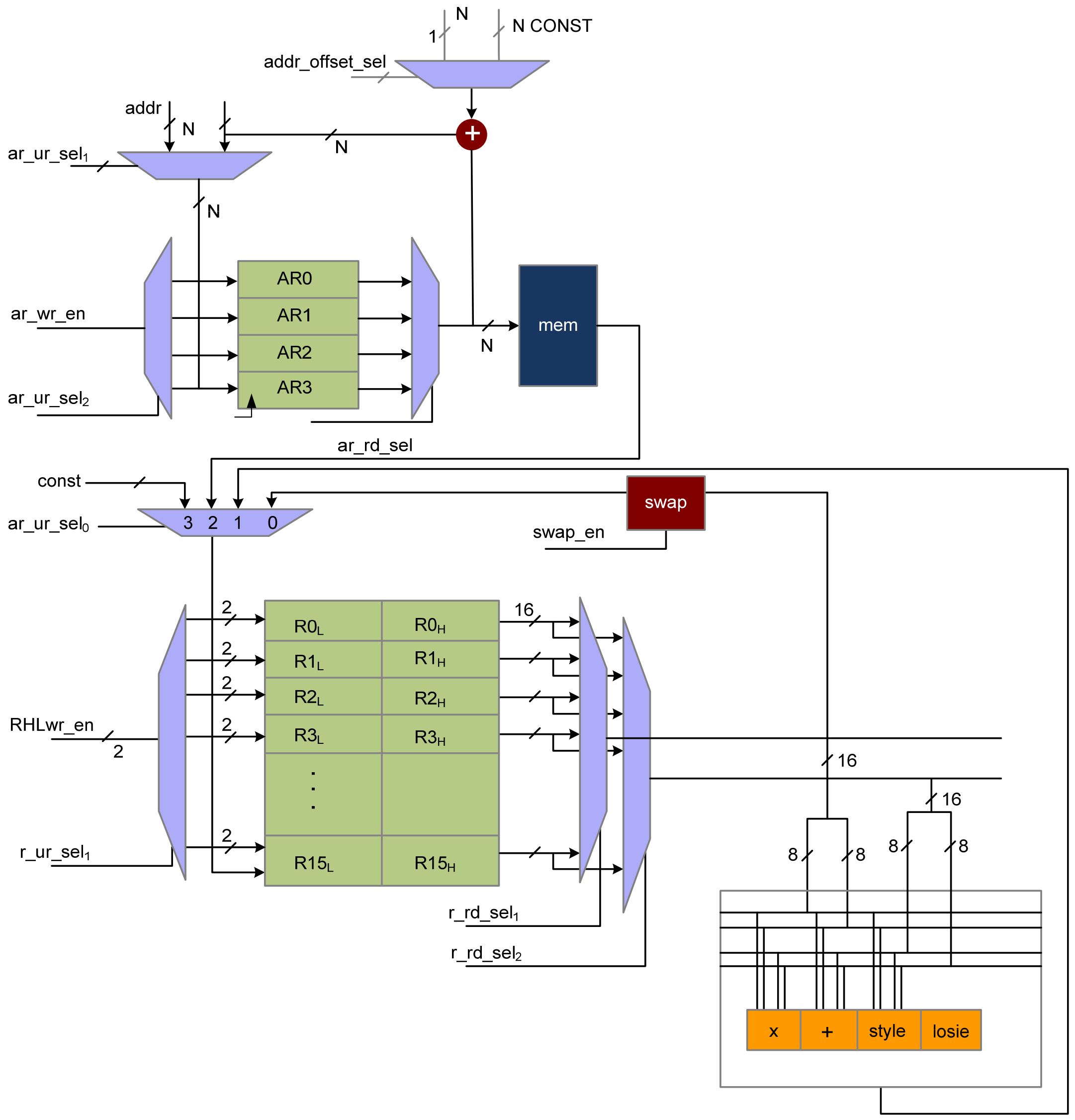
(b)

(a) One level 2-D wavelet decomposition
(b) Three level 2-D wavelet decomposition

Example: Design of a Wavelet Processor



Example: Design of a Wavelet Processor



Questions/Feedback