Mic-1: Microarchitecture

University of Fribourg, Switzerland

System I: Introduction to

Computer Architecture

WS 2005-2006

20 December 2006

Béat Hirsbrunner, Amos Brocco, Fulvio Frapolli

Mic-1: Microarchitecture (1)

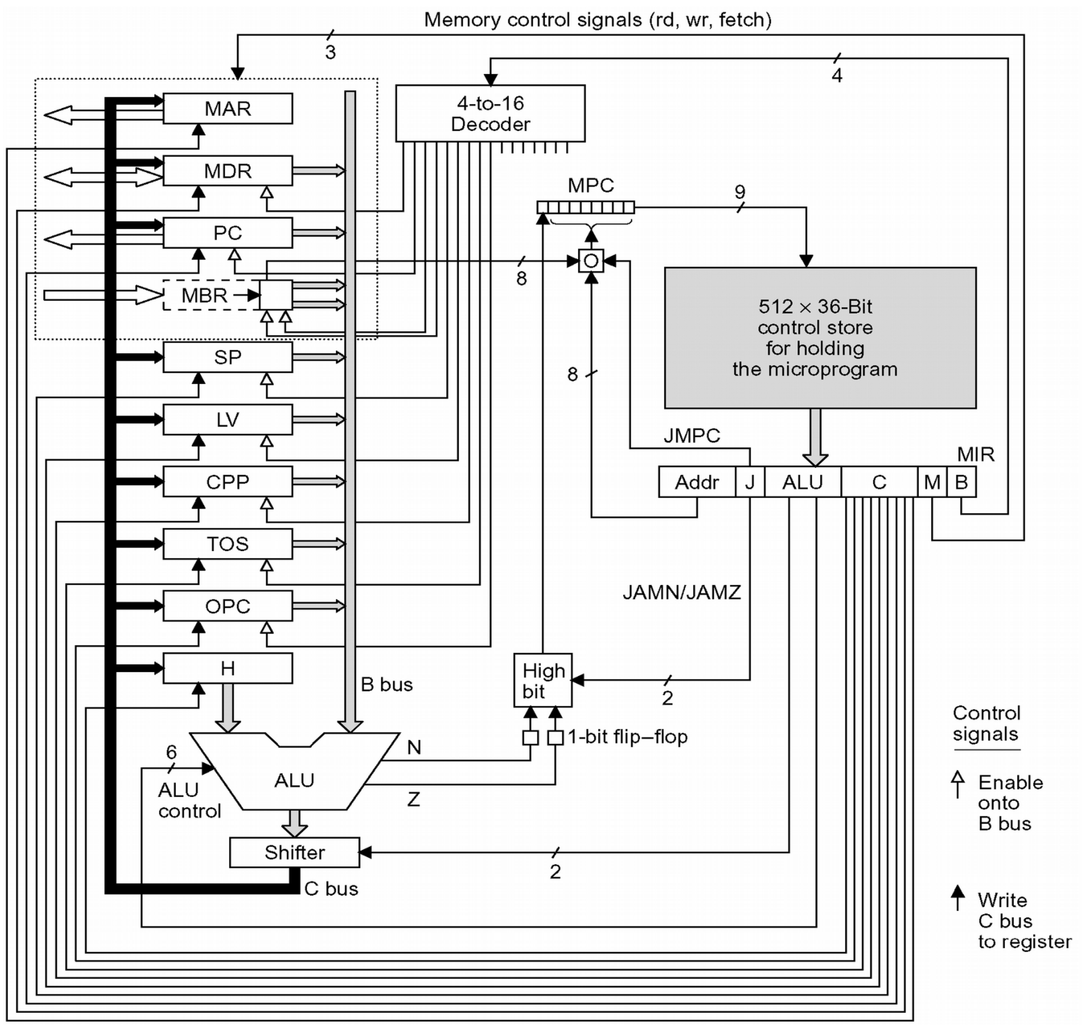


Figure 4-6. The complete block diagram of our example microarchitecture, the Mic-1.

Mic-1: Microarchitecture (2)

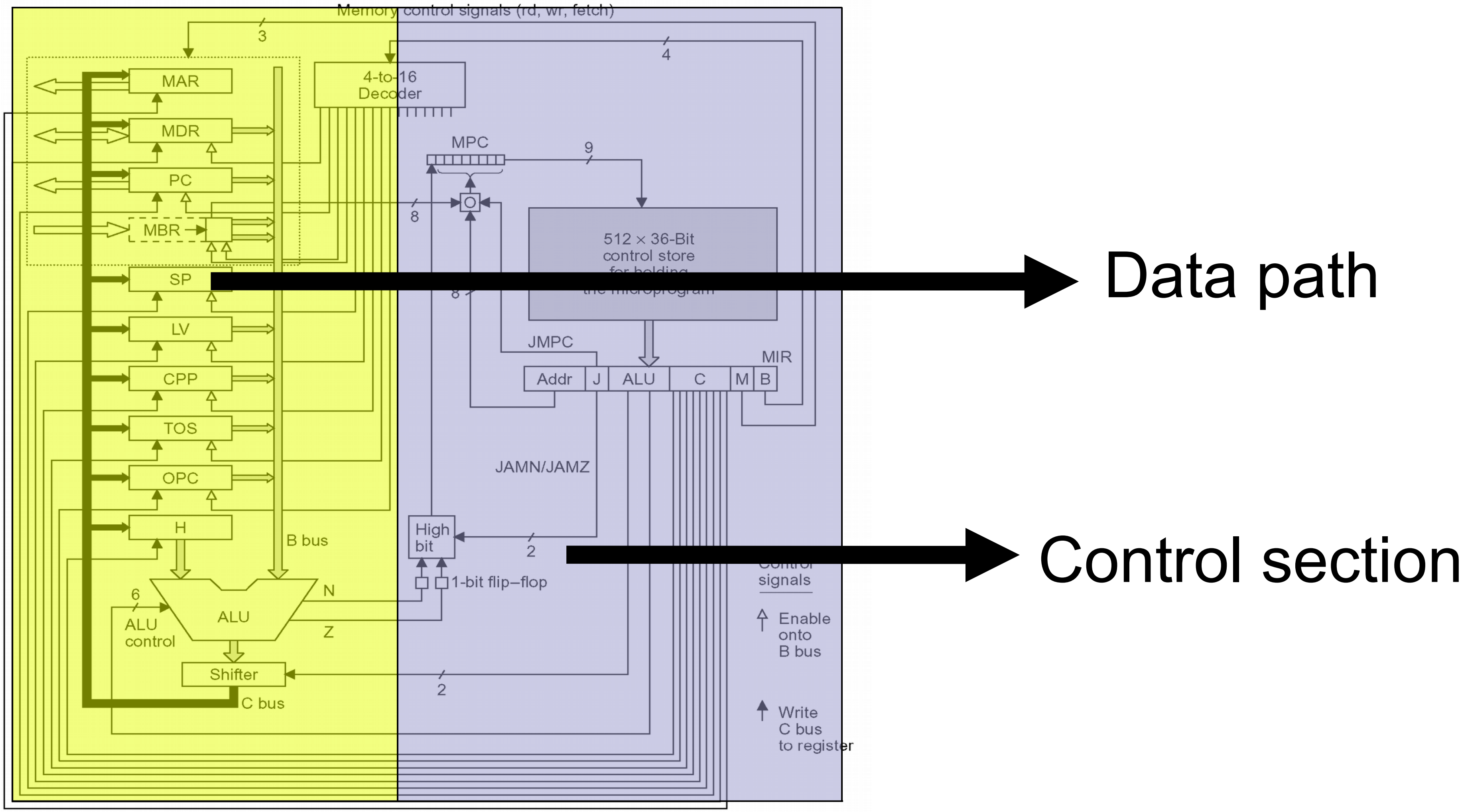
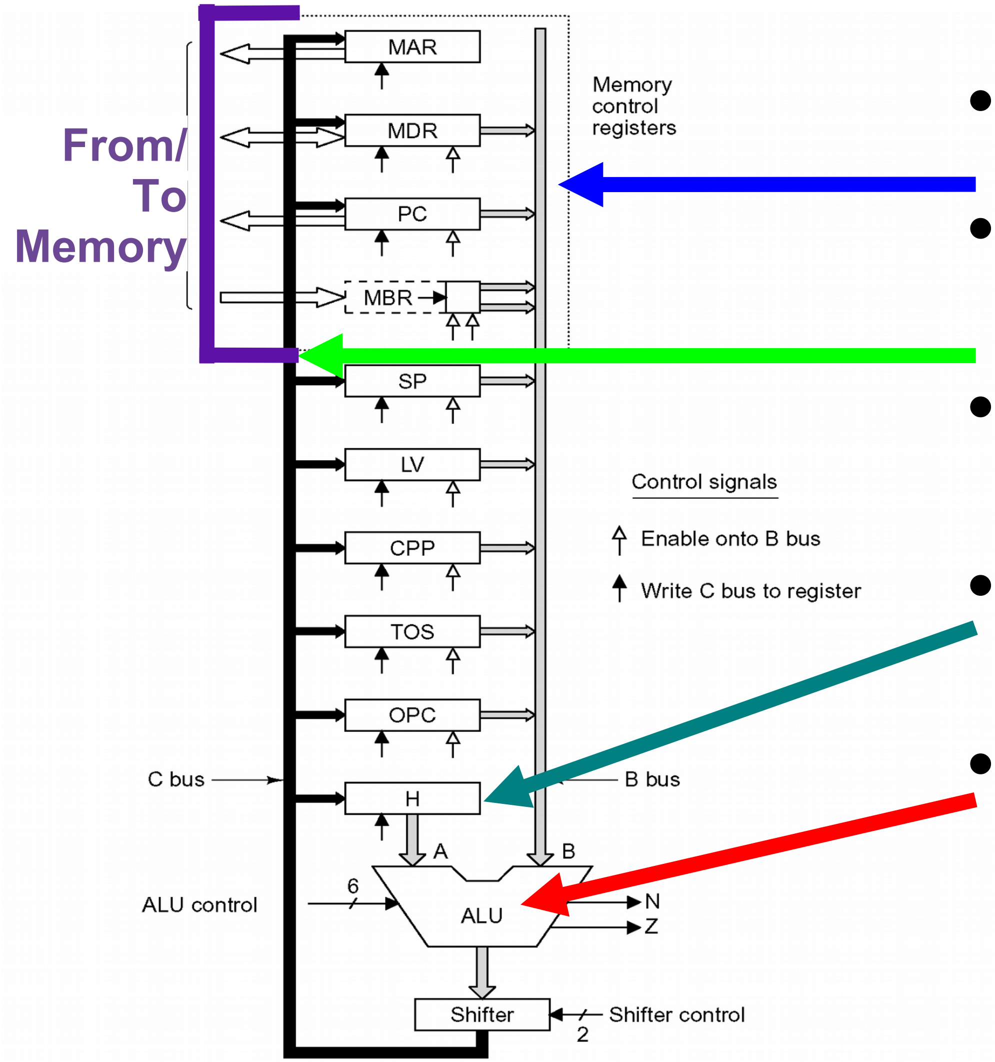


Figure 4-6. The complete block diagram of our example microarchitecture, the Mic-1.

The data path



- 32-bit registers (with exception of MBR, which is a 8 bit register)
- **B** bus to drive data to the ALU
- **C** bus to drive data from the ALU to registers
- **H** register as A-input of the ALU
- ALU with 6 **control signals** (and 2 outputs, N to test for Negative numbers and Z to test for Zero)

Figure 4-1. The data path of the example microarchitecture used in this chapter.

ALU Control Signals

F_0	F_1	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\overline{A}$
1	0	1	1	0	0	$\overline{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B – A
1	1	0	1	1	1	B – 1
1	1	1	0	1	1	–A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
0	1	0	0	1	0	–1

Figure 4-2. Useful combinations of ALU signals and the function performed.

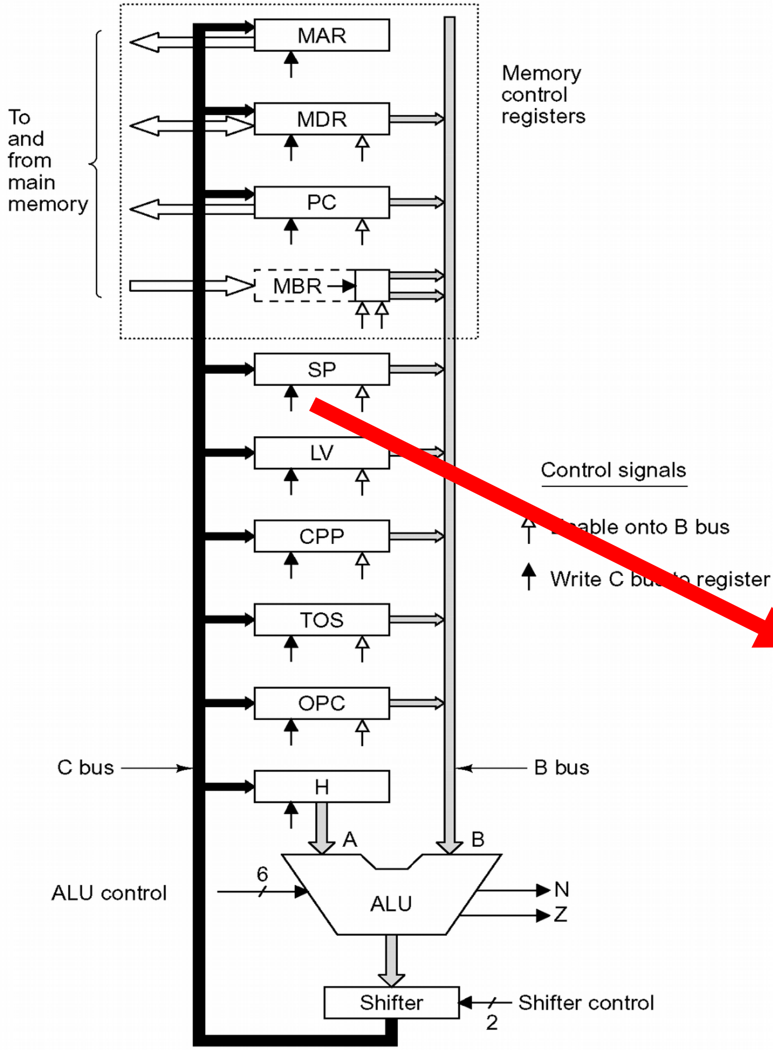
The data path

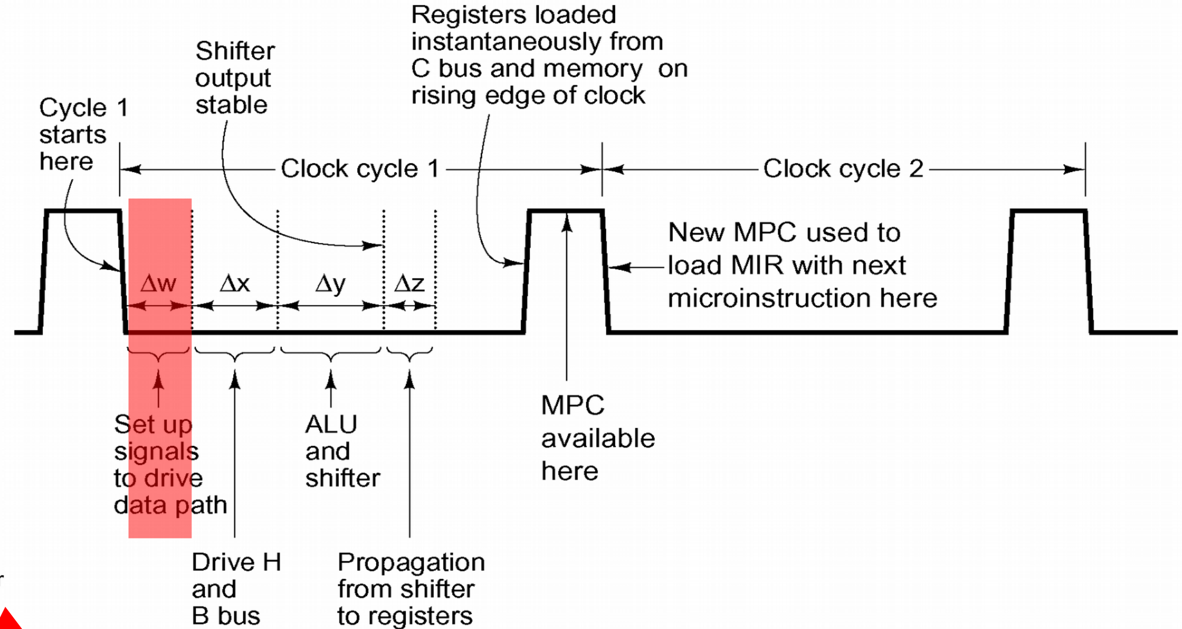
Registers have control signals to enable/disable **reading** from them (put value on the B bus) and **writing** to them (store value from the C bus)

It is possible to **read only from one register at time**: so we can use a 4 -> 16 bit decoder

It is possible to **write to one or more registers at the same time**: so we need 9 control signals for the C bus.

Data path synchronization (1)

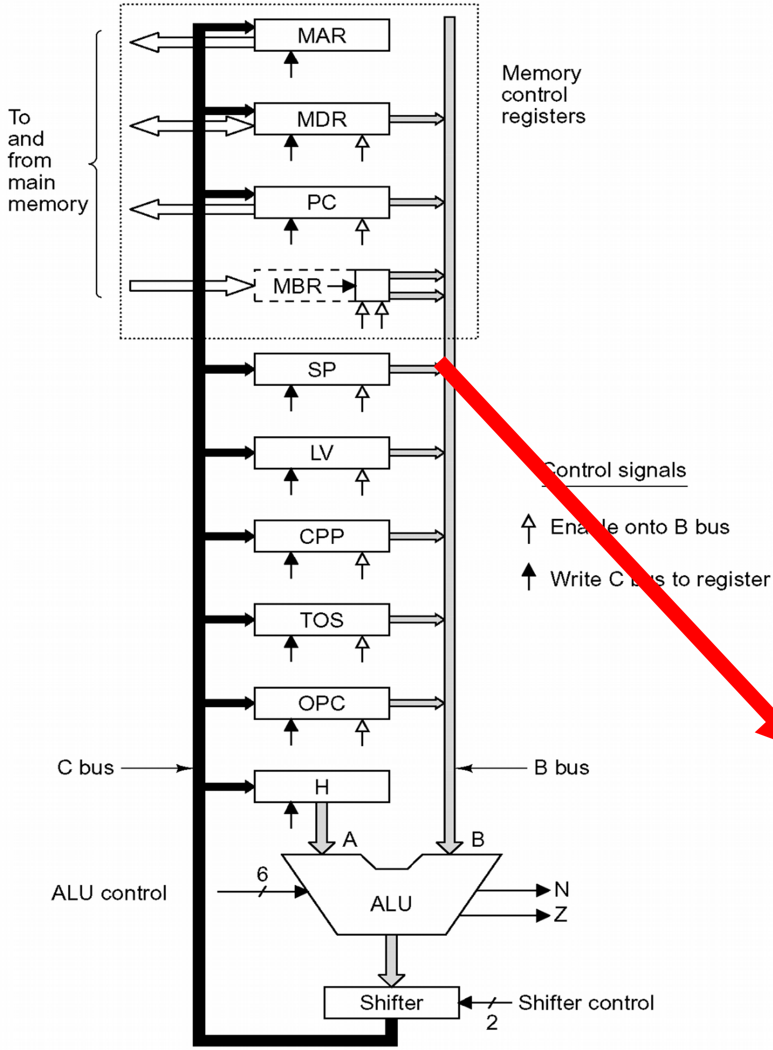


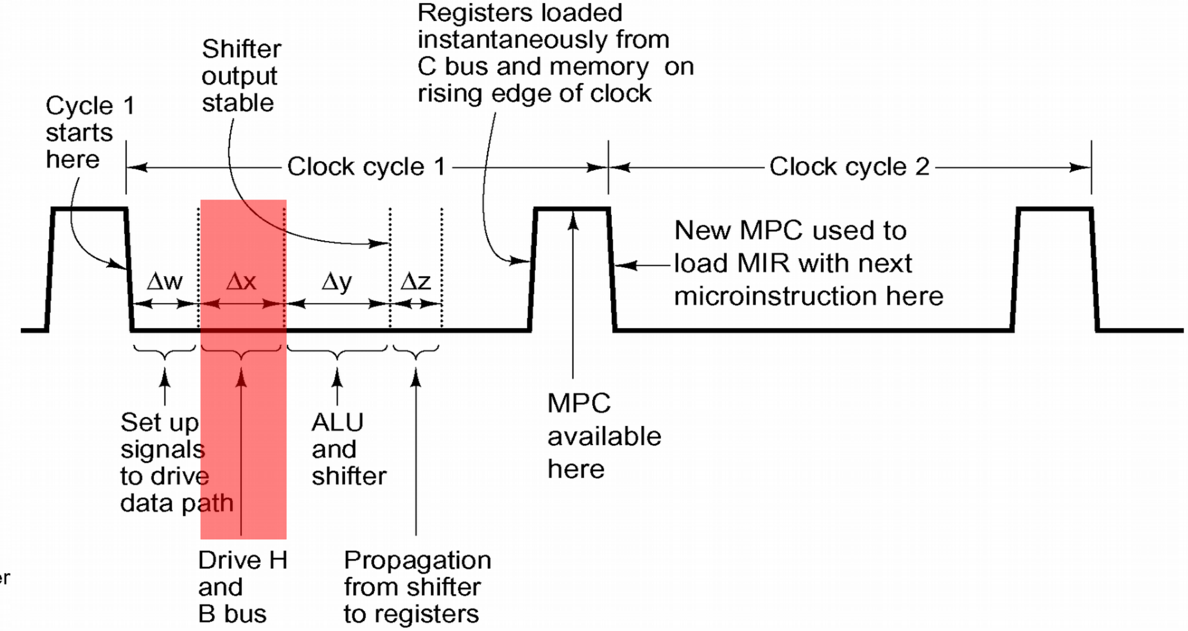


1. Control signals stabilize
2. A register value is put on the B bus
3. ALU and shifter operate
4. Result propagate on the C bus
5. Result is written in the registers on the raising edge of the next clock pulse

Figure 4-1. The data path of the example microarchitecture used in this chapter.

Data path synchronization (2)

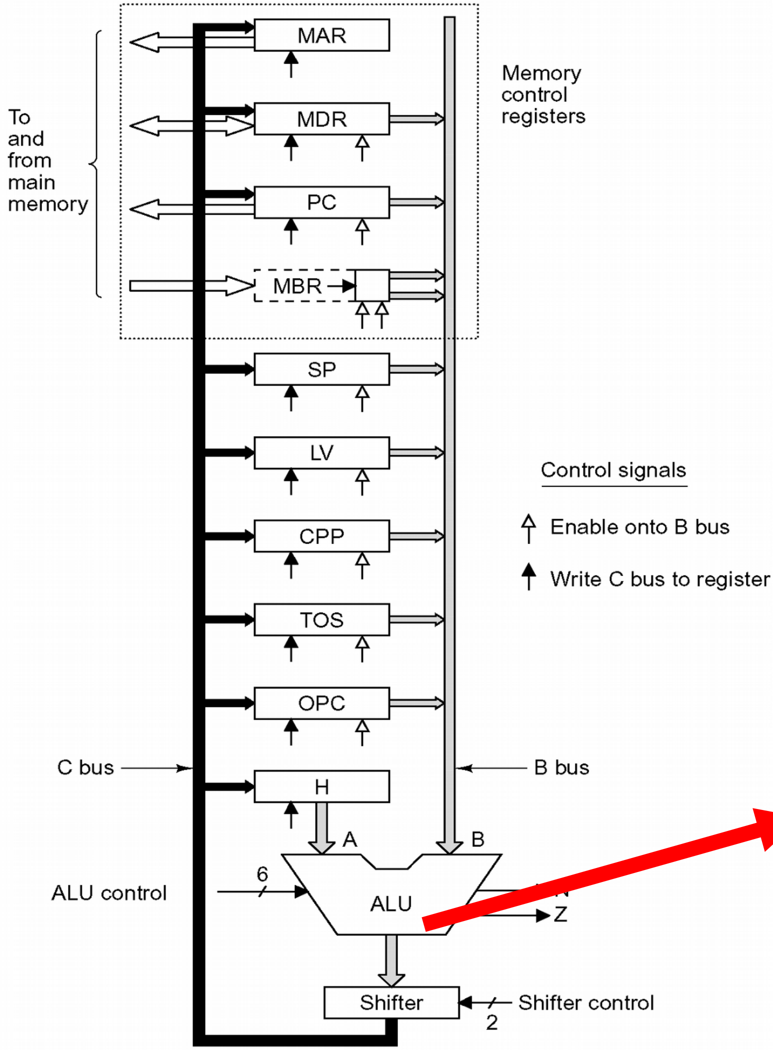


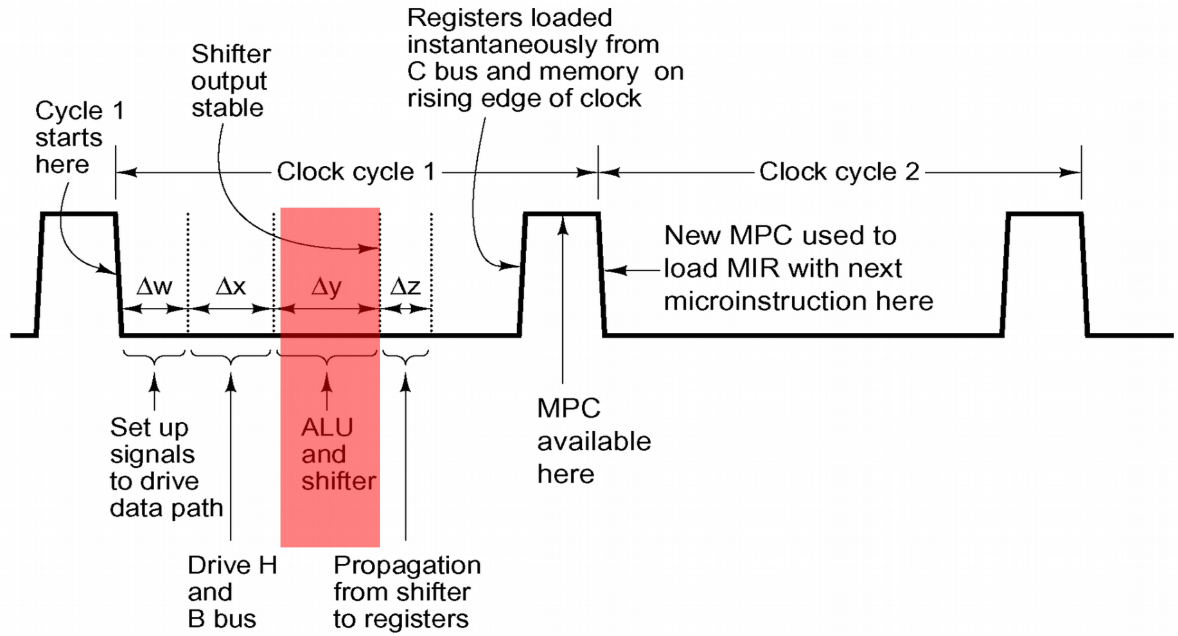


1. Control signals stabilize
2. Register's value is put on the B bus
3. ALU and shifter operate
4. Result propagate on the C bus
5. Result is written in the registers on the raising edge of the next clock pulse

Figure 4-1. The data path of the example microarchitecture used in this chapter.

Data path synchronization (3)

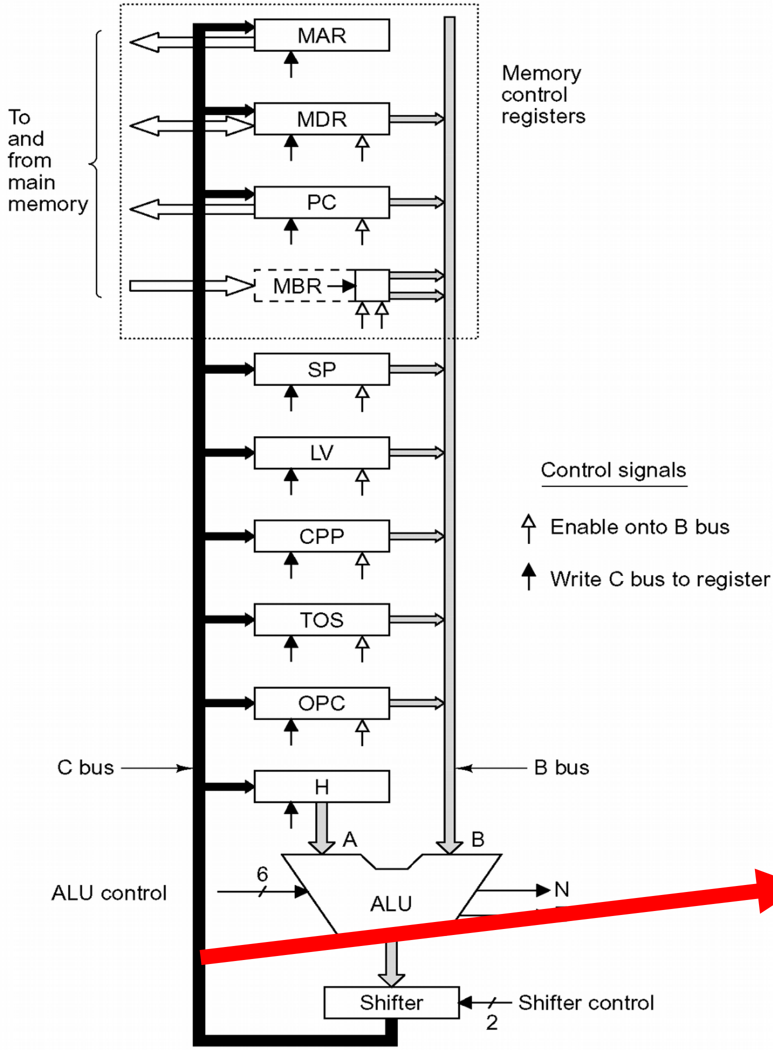


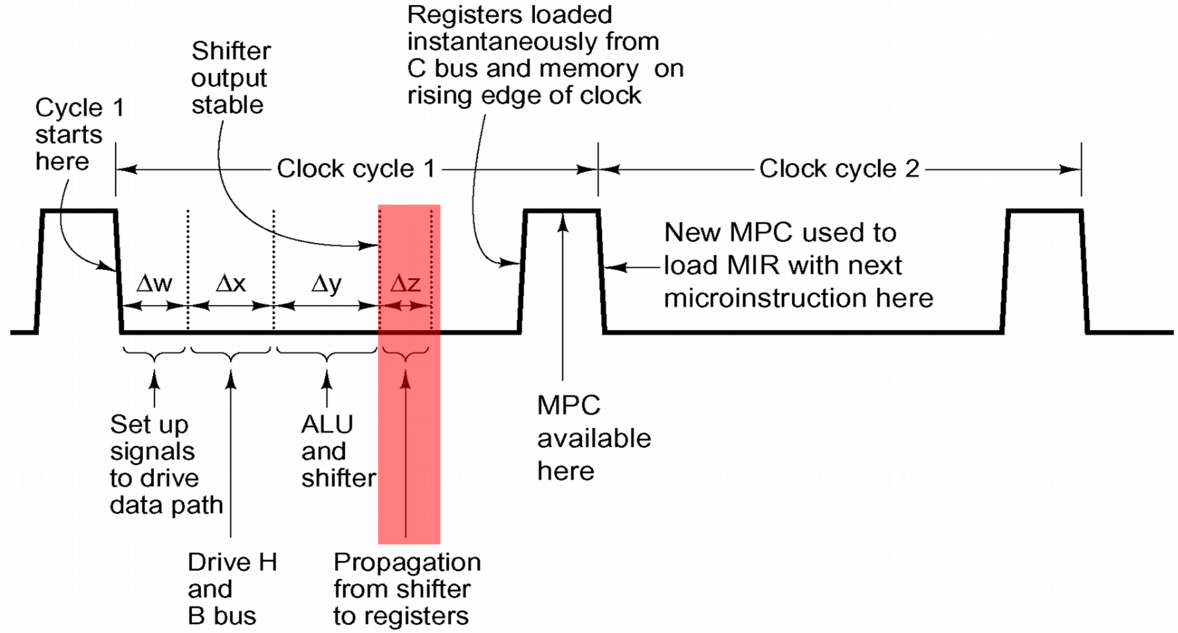


1. Control signals stabilize
2. Register's value is put on the B bus
3. ALU and shifter operate
4. Result propagate on the C bus
5. Result is written in the registers on the raising edge of the next clock pulse

Figure 4-1. The data path of the example microarchitecture used in this chapter.

Data path synchronization (4)

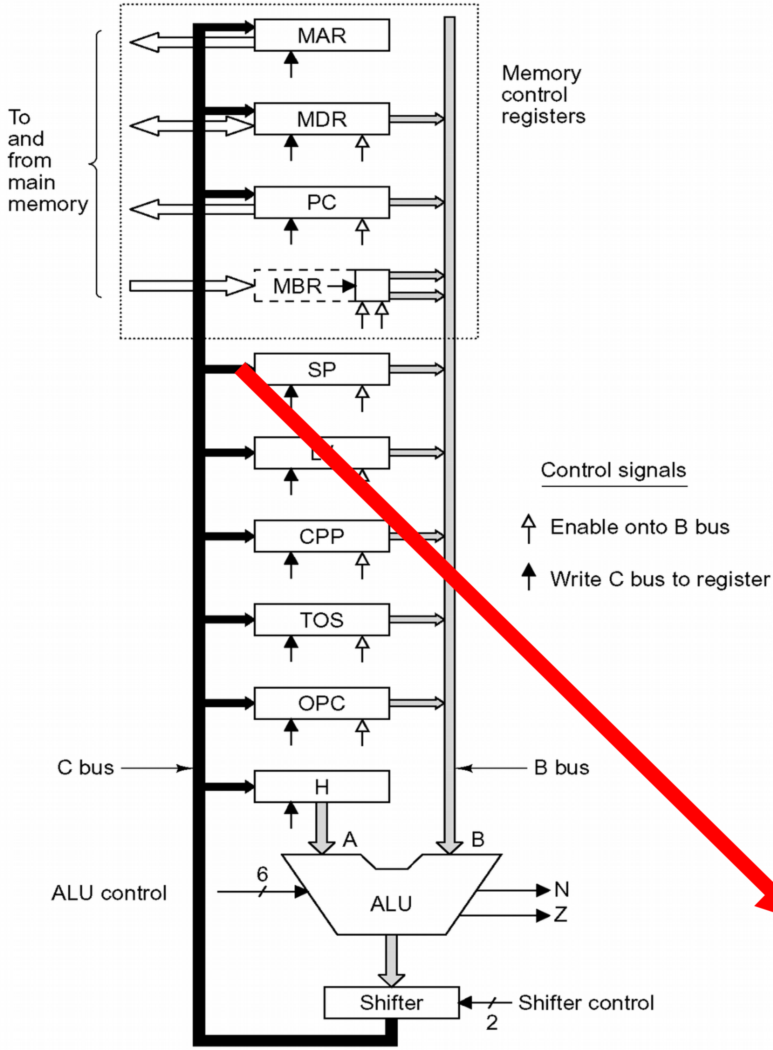


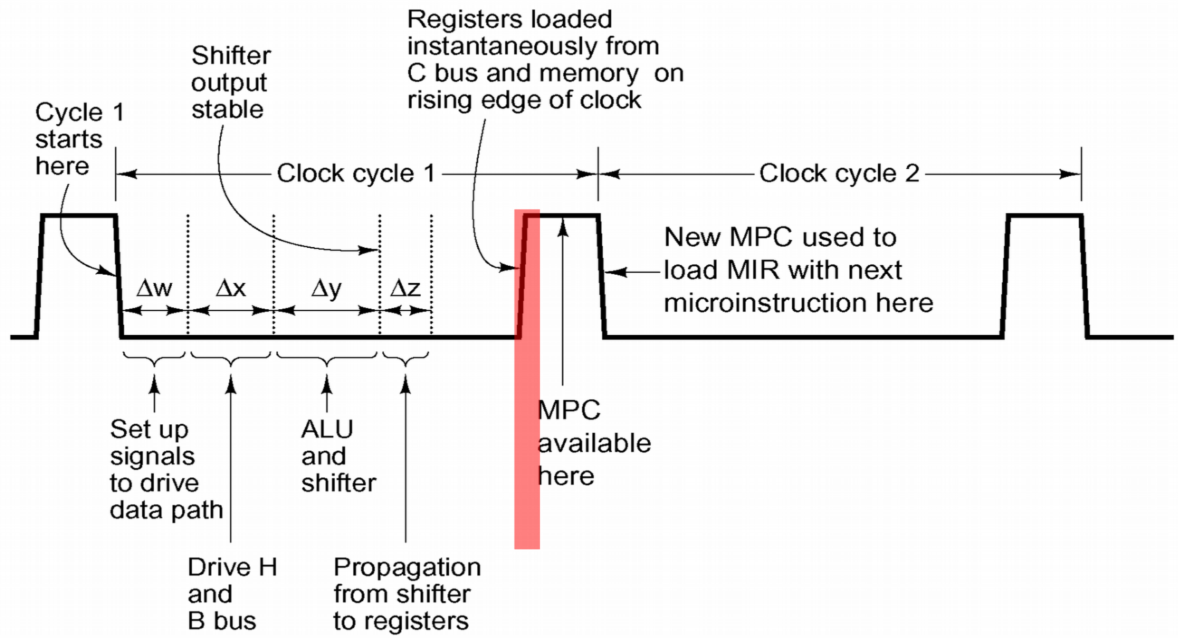


1. Control signals stabilize
2. Register's value is put on the B bus
3. ALU and shifter operate
4. Result propagate on the C bus
5. Result is written in the registers on the raising edge of the next clock pulse

Figure 4-1. The data path of the example microarchitecture used in this chapter.

Data path synchronization (4)

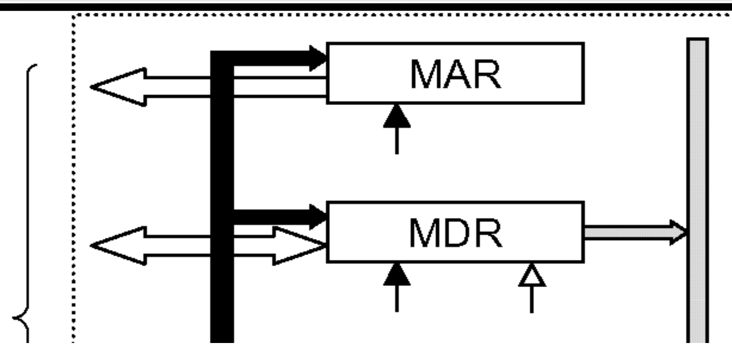




1. Control signals stabilize
2. Register's value is put on the B bus
3. ALU and shifter operate
4. Result propagate on the C bus
5. Result is written in the registers on the raising edge of the next clock pulse

Figure 4-1. The data path of the example microarchitecture used in this chapter.

MAR and MDR (1)



32 bit registers connected to the main memory

MAR = **M**emory **A**ddress **R**egister

MDR = **M**emory **D**ata **R**egister

MAR has **only one** control signal (input from C)

Two memory operations: **read** and **write**

MAR and MDR (2)

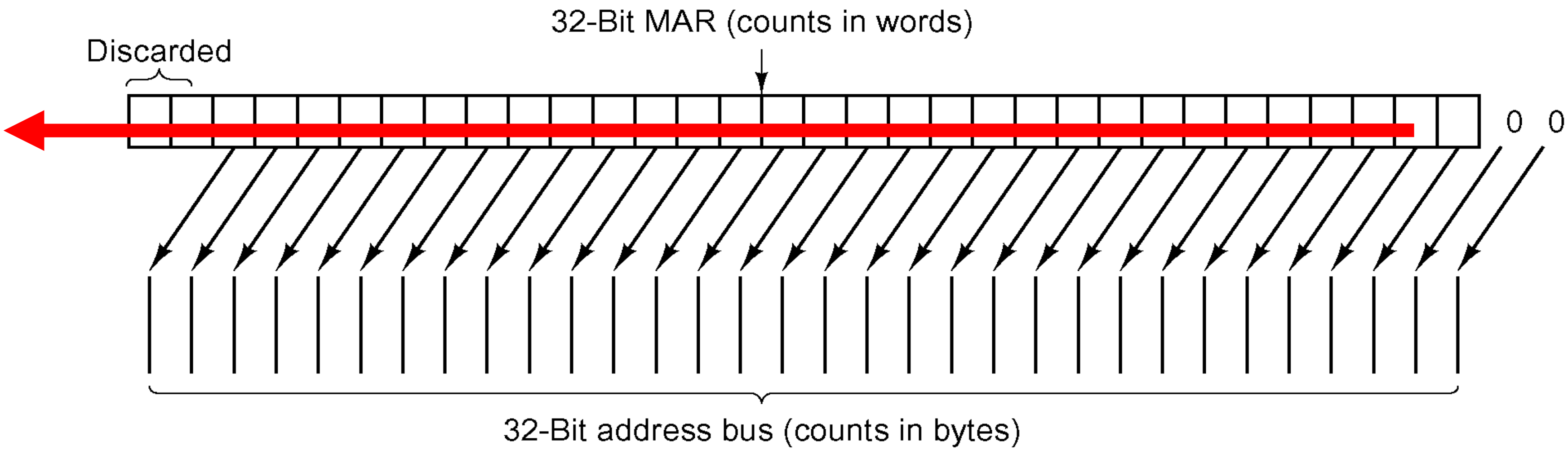


Figure 4-4. Mapping of the bits in MAR to the address bus.

Data is **word** (4*8bit = 32bit in our ISA) addressed!

=>MAR addresses are **shifted 2bit left** (= * 4)

Memory Access

A memory read initiated at cycle **k** delivers data that can be used only in cycle **k+2** or later!

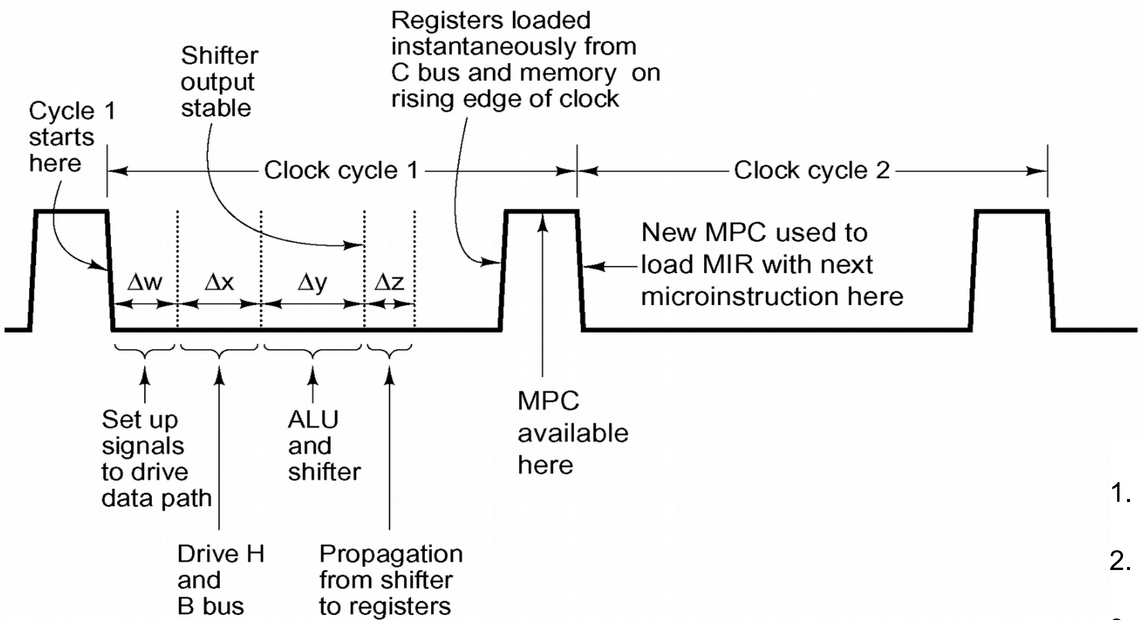


Figure 4-3. Timing diagram of one data path cycle.

1. MAR is loaded
2. Memory access
3. MDR is loaded with data read from memory
4. Data in MDR is available

Memory Access

A memory read initiated at cycle **k** delivers data that can be used only in cycle **k+2** or later!

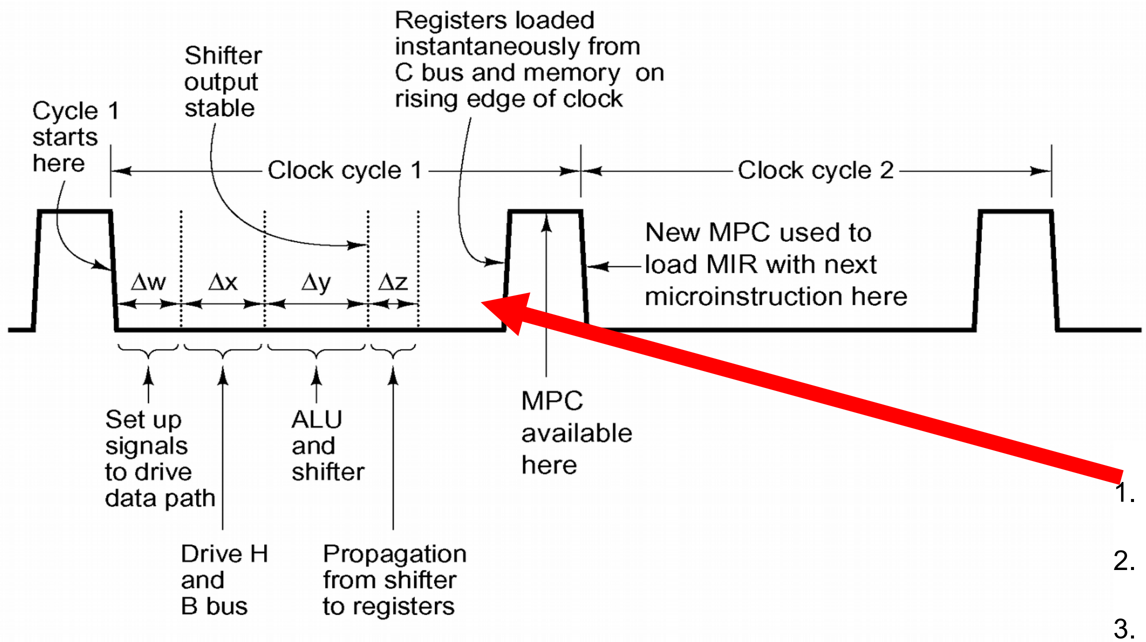


Figure 4-3. Timing diagram of one data path cycle.

1. MAR is loaded
2. Memory access
3. MDR is loaded with data read from memory
4. Data in MDR is available

Memory Access

A memory read initiated at cycle **k** delivers data that can be used only in cycle **k+2** or later!

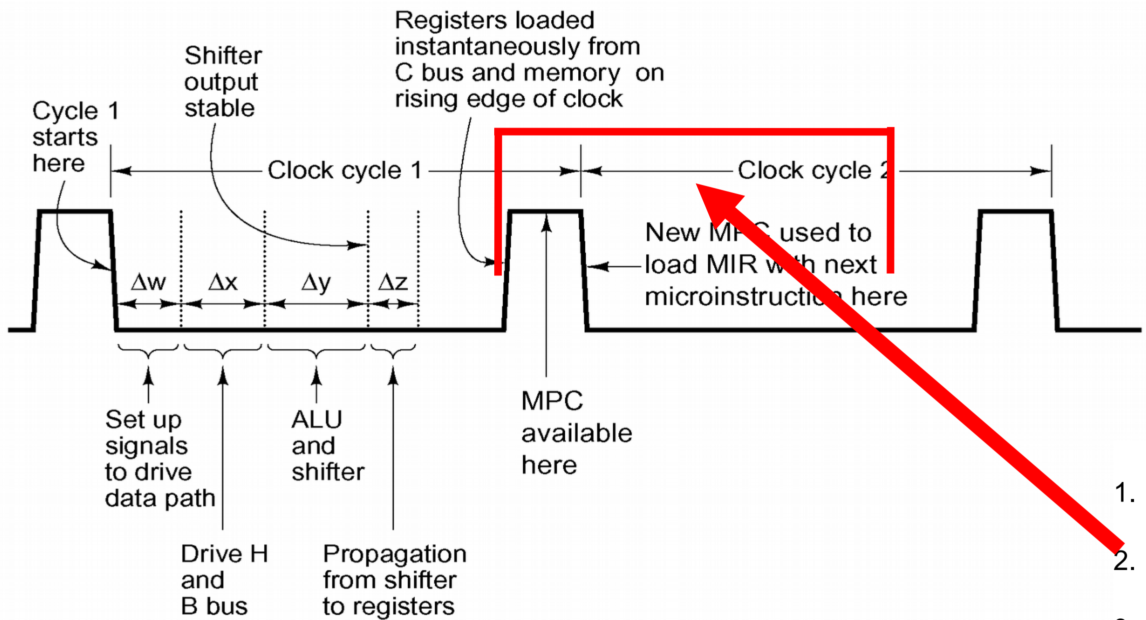


Figure 4-3. Timing diagram of one data path cycle.

1. MAR is loaded
2. Memory access
3. MDR is loaded with data read from memory
4. Data in MDR is available

Memory Access

A memory read initiated at cycle **k** delivers data that can be used only in cycle **k+2** or later!

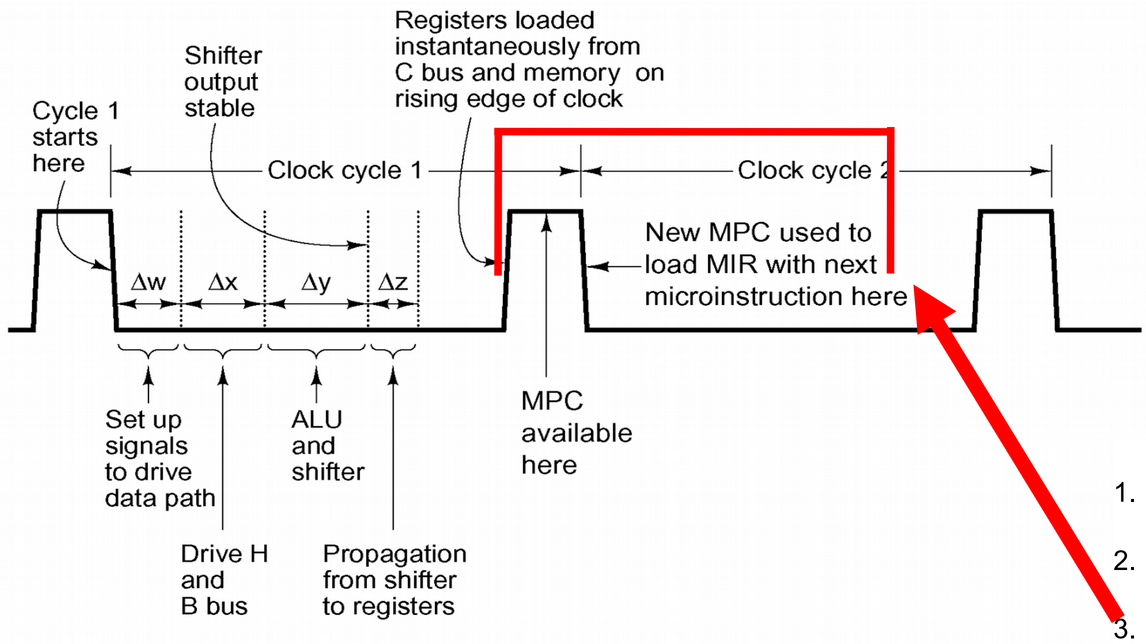


Figure 4-3. Timing diagram of one data path cycle.

1. MAR is loaded
2. Memory access
3. MDR is loaded with data read from memory
4. Data in MDR is available

Memory Access

A memory read initiated at cycle **k** delivers data that can be used only in cycle **k+2** or later!

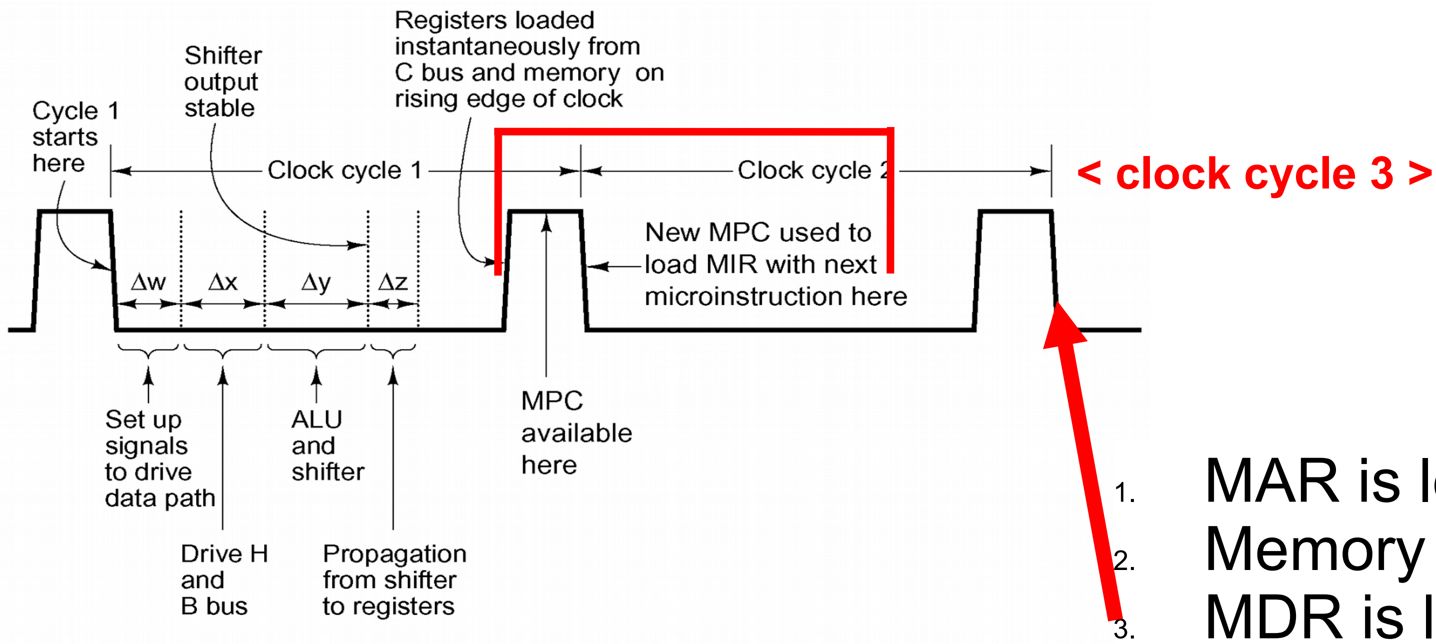


Figure 4-3. Timing diagram of one data path cycle.

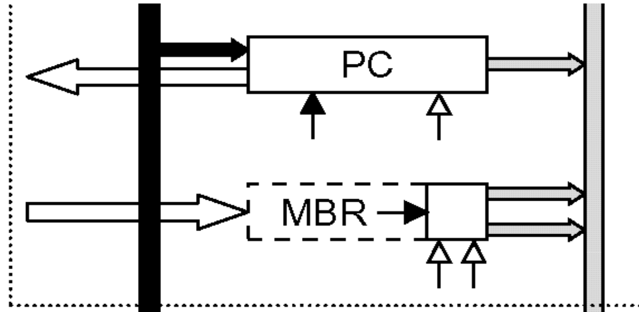
1. MAR is loaded
2. Memory access
3. MDR is loaded with data read from memory
4. Data in MDR is available

Memory Access (2)

Until start of cycle k+2 the MDR register **contains old data**

It is possible to issue **consecutive requests**, for example at time **k** and **k+1**: corresponding results will be available at **k+2** and **k+3**

PC and MBR



8 bit registers connected to the main memory used to read (fetch) ISA instructions

PC = **P**rogram **C**ounter

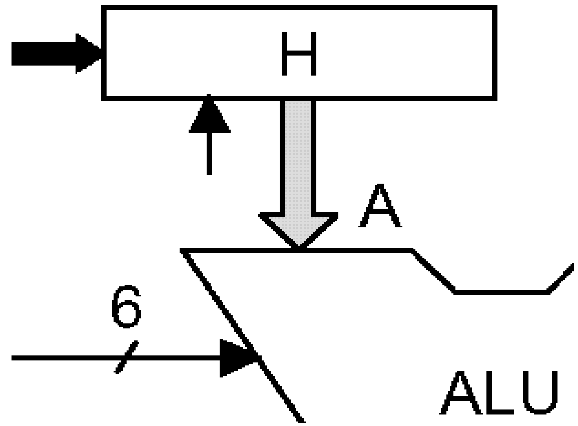
MBR = **M**emory **B**uffer **R**egister

Access also requires one clock cycle (k -> k+2)

MBR has two control signals for the B bus, for **signed** or **unsigned** operations

One memory operation: **fetch**

H register



Is the A-input of the ALU

Has **only one control signal**; output to the ALU is always enabled

ISA, IJVM, Microarchitecture

ISA = Instruction Set Architecture
(defines instructions, memory model, available registers,...)

IJVM = An example ISA (it's stack based architecture)

The **IJVM** (Integer Java Virtual Machine) level executes the **IJVM Instruction set**

The IJVM is (in this case) implemented by the **Mic-1 Microarchitecture**

Mic-1 implementation

The Mic-1 is a microprogrammed architecture: each IJVM instruction (**Macroinstruction**) is divided one or more steps.

In each step, a **microinstruction** is executed by the Mic-1.

Microinstructions are simpler than ISA macroinstructions.

Control section

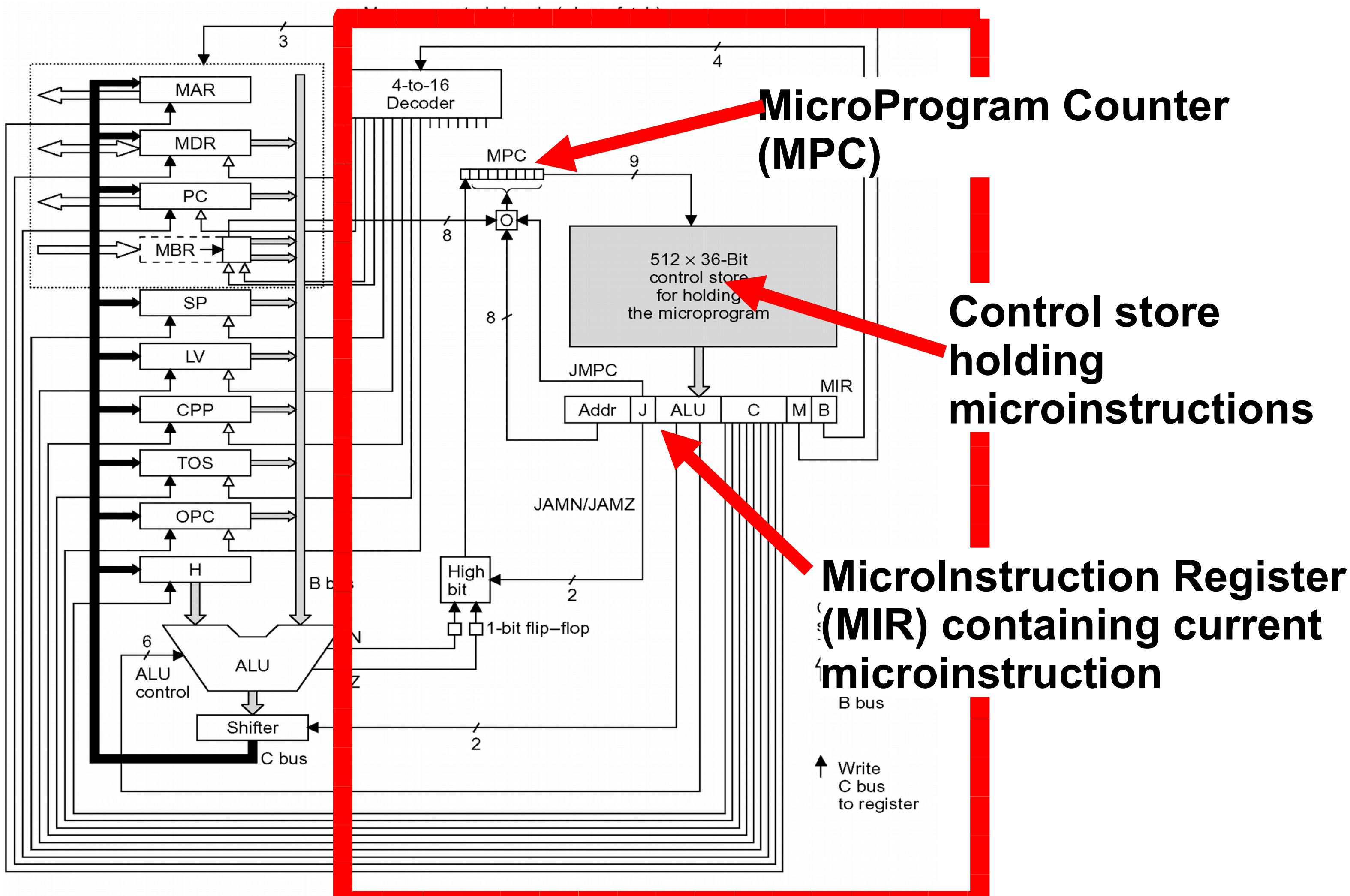


Figure 4-6. The complete block diagram of our example microarchitecture, the Mic-1.

Microinstructions

36bit wide microinstructions

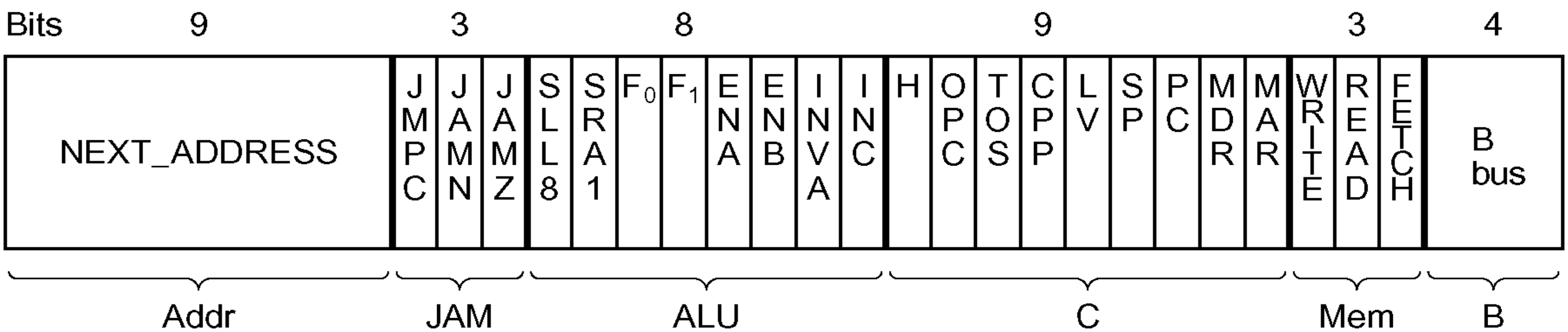
Microinstructions are “executed” in the **control section** (“a CPU in the CPU”)

Microinstructions basically drive **control signals** for the data path.

To avoid the need for a real (micro)Program Counter **each microinstruction specifies the address of the following one**.

Microinstruction addresses are 9-bit wide

Microinstruction format (1)



B bus registers

0 = MDR
1 = PC
2 = MBR
3 = MBRU
4 = SP
5 = LV
6 = CPP
7 = TOS
8 = OPC
9-15 none

Figure 4-5. The microinstruction format for the Mic-1.

Microinstruction format (2)

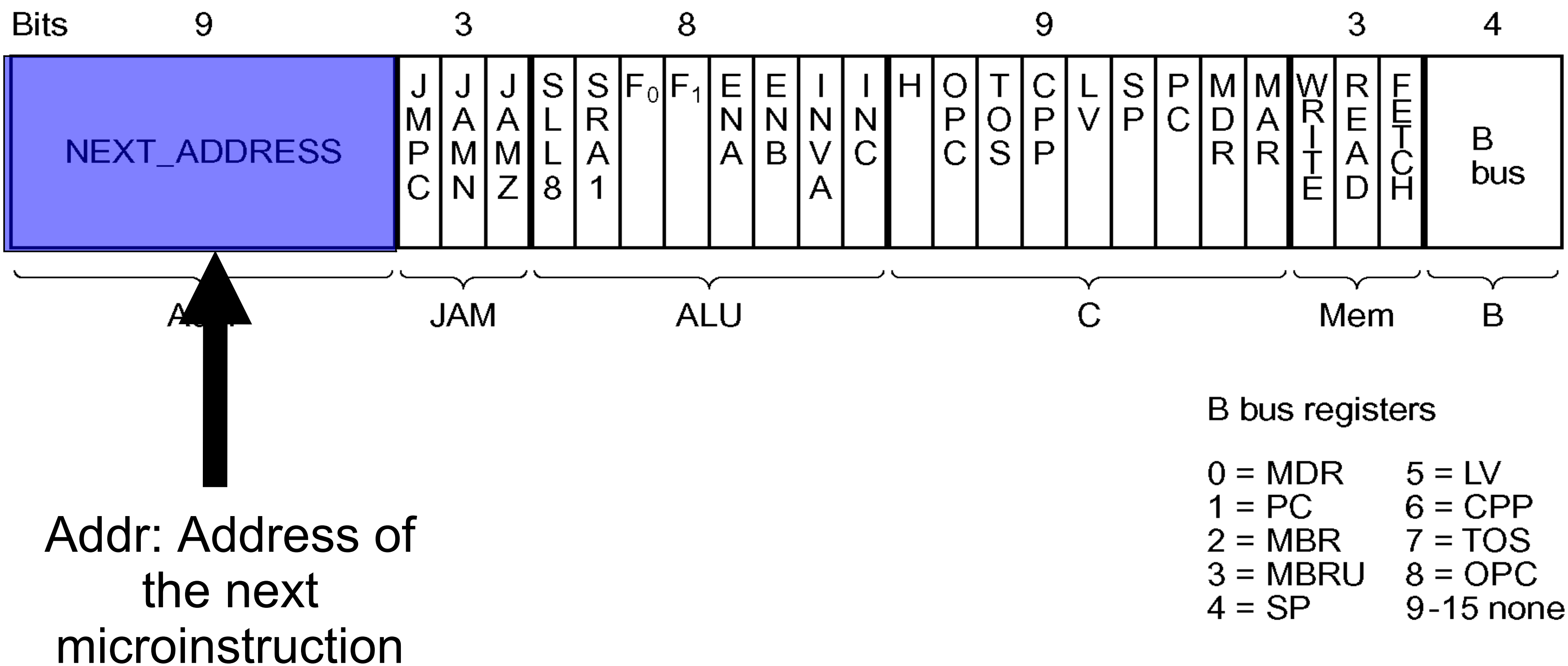
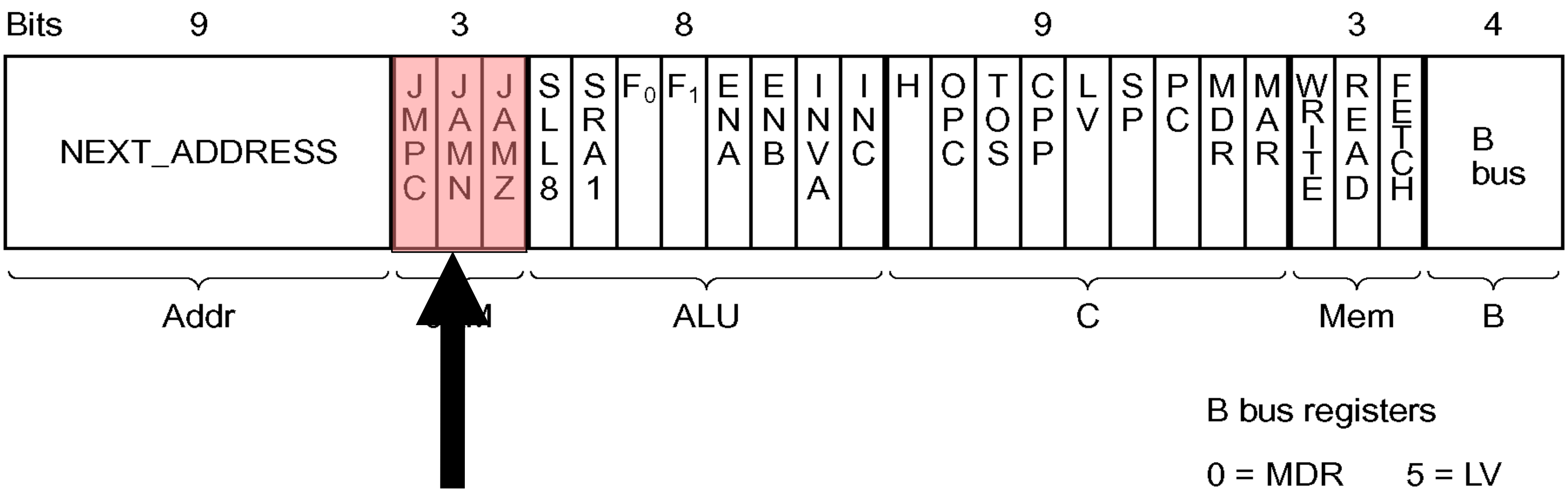


Figure 4-5. The microinstruction format for the Mic-1.

Microinstruction format (3)



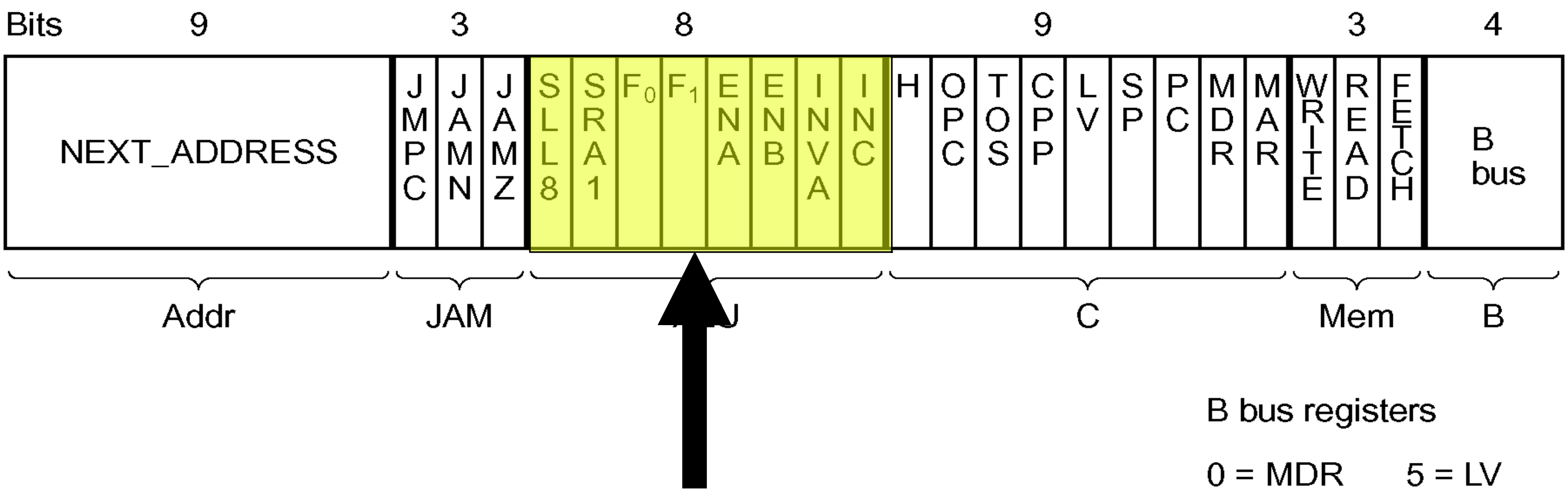
JAM: Determines how to choose next microinstruction

B bus registers

0 = MDR
1 = PC
2 = MBR
3 = MBRU
4 = SP
5 = LV
6 = CPP
7 = TOS
8 = OPC
9-15 none

Figure 4-5. The microinstruction format for the Mic-1.

Microinstruction format (4)



ALU: Control signals to choose ALU operations

B bus registers

0 = MDR	5 = LV
1 = PC	6 = CPP
2 = MBR	7 = TOS
3 = MBRU	8 = OPC
4 = SP	9-15 none

Figure 4-5. The microinstruction format for the Mic-1.

Microinstruction format (5)

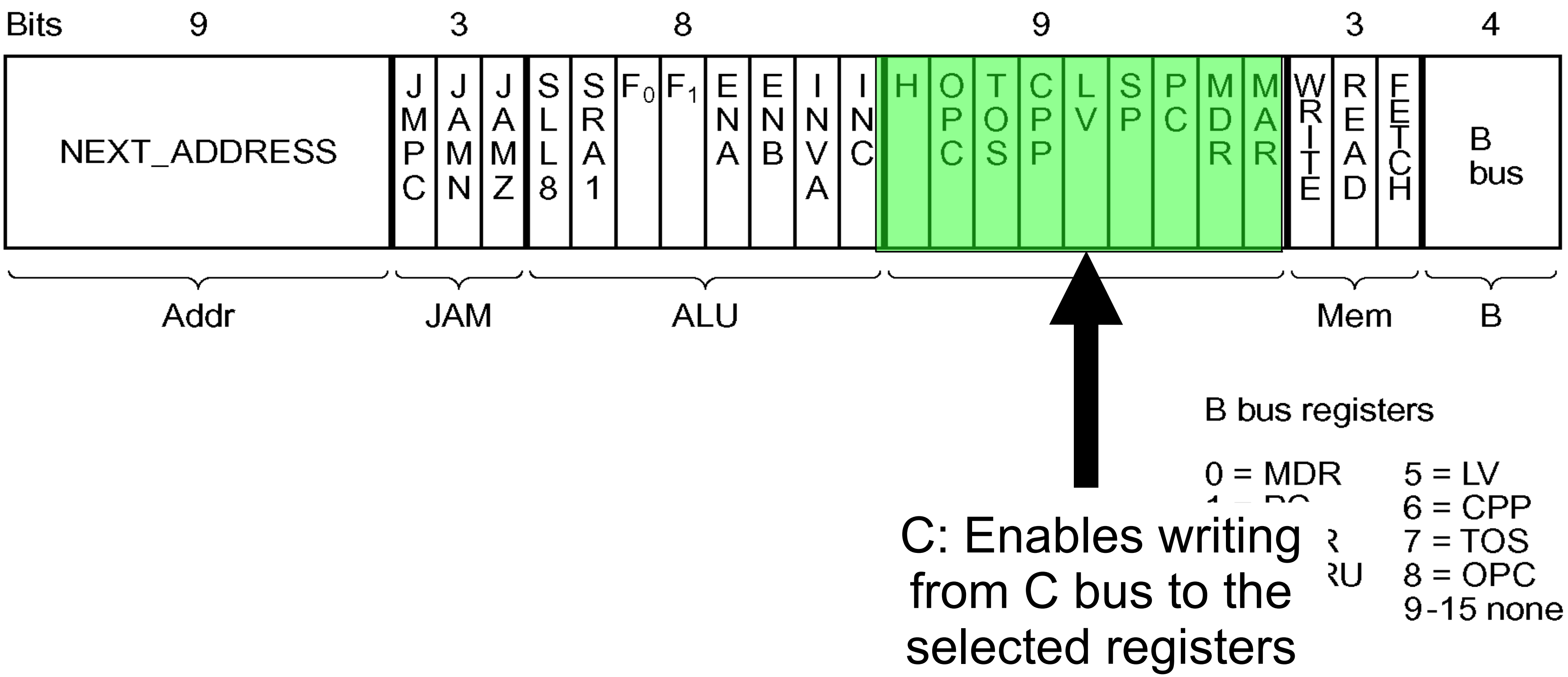
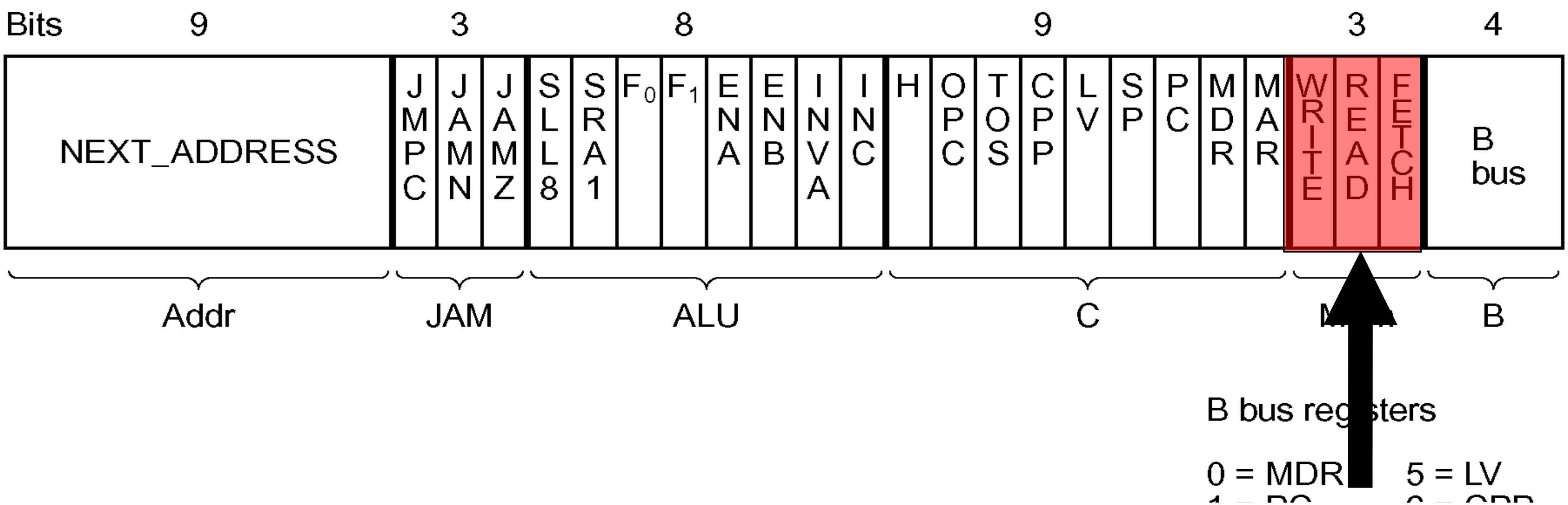


Figure 4-5. The microinstruction format for the Mic-1.

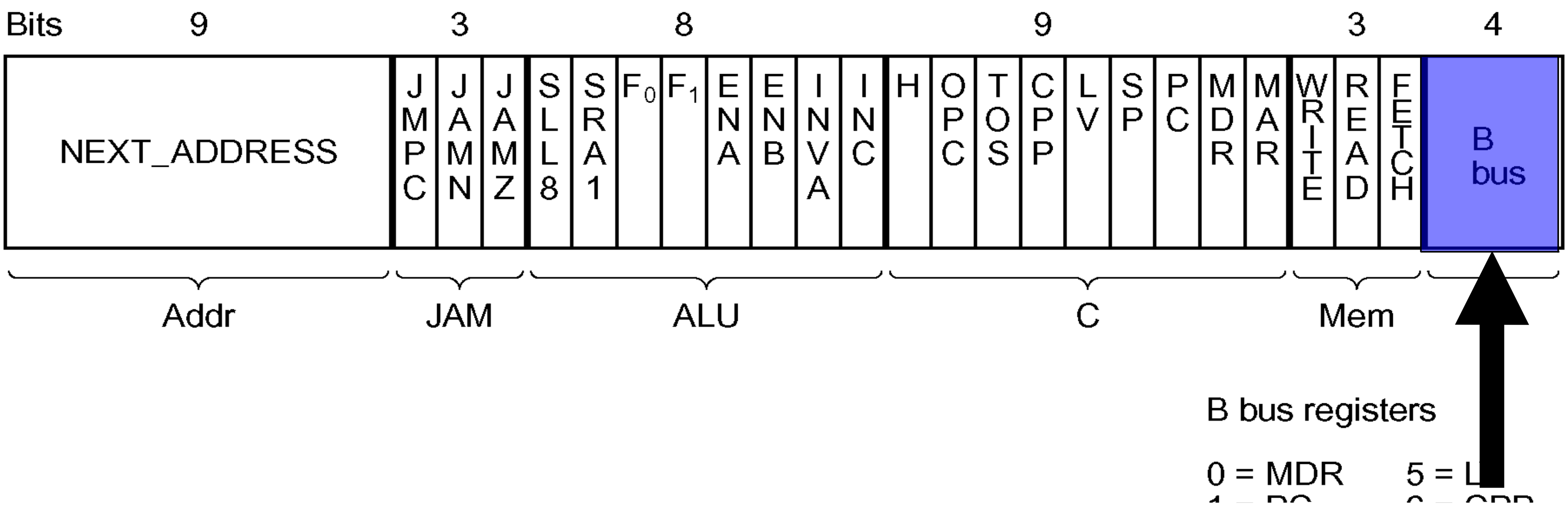
Microinstruction format (6)



Mem: Controls memory read/write/fetch operations

Figure 4-5. The microinstruction format for the Mic-1.

Microinstruction format (7)



B: Controls which register can write to the B bus

Figure 4-5. The microinstruction format for the Mic-1.

Driving control signals

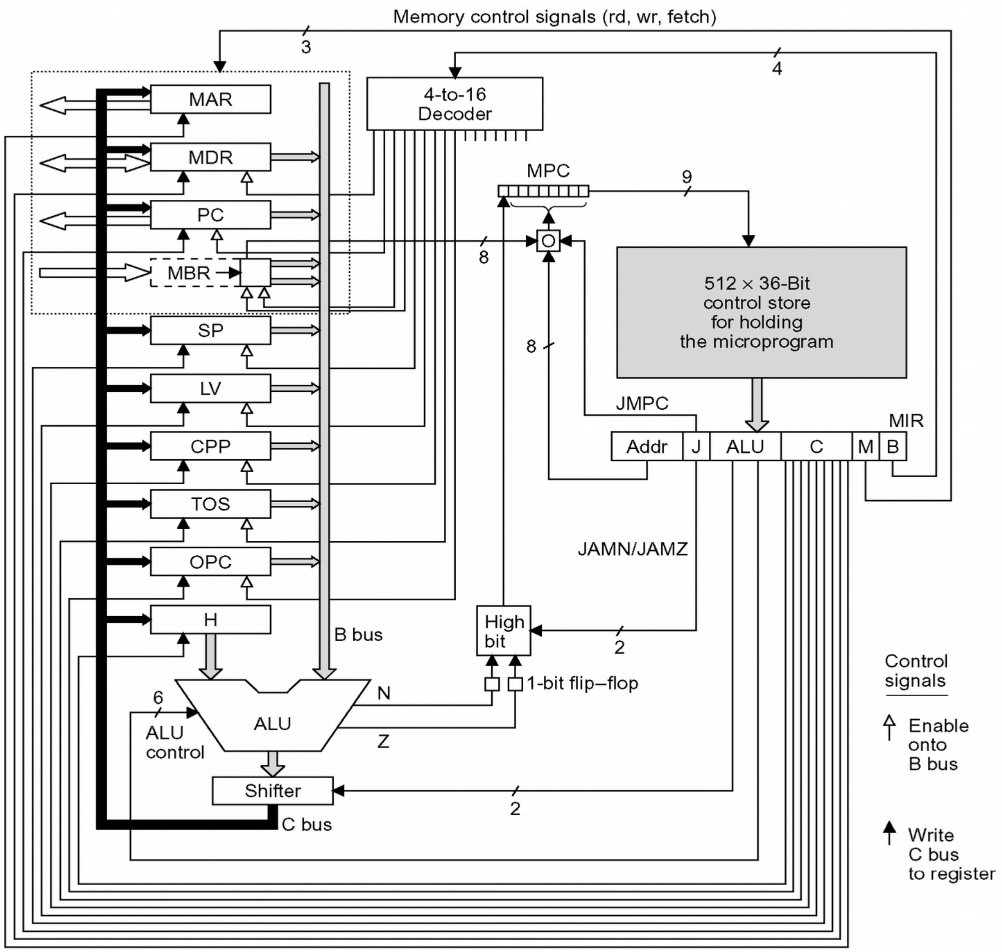


Figure 4-6. The complete block diagram of our example microarchitecture, the Mic-1.

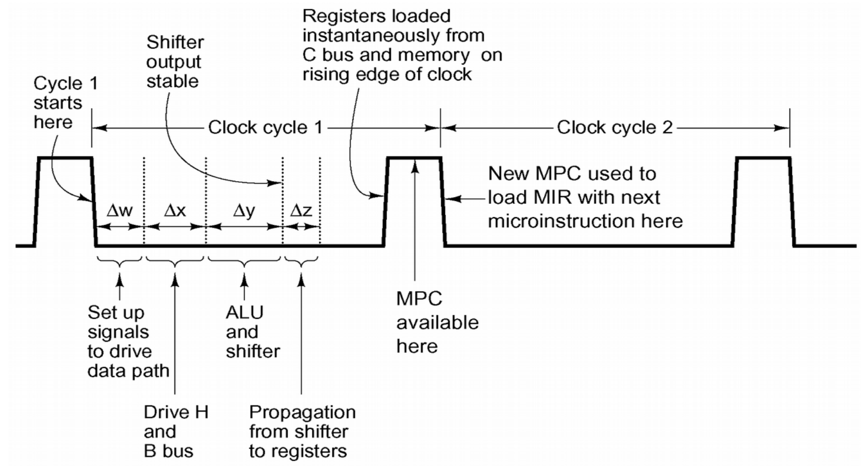


Figure 4-3. Timing diagram of one data path cycle.

- MIR is loaded on the falling edge of the clock based on the MPC address, control signals propagate

1. ALU Operation: N and Z values available and saved

Driving control signals

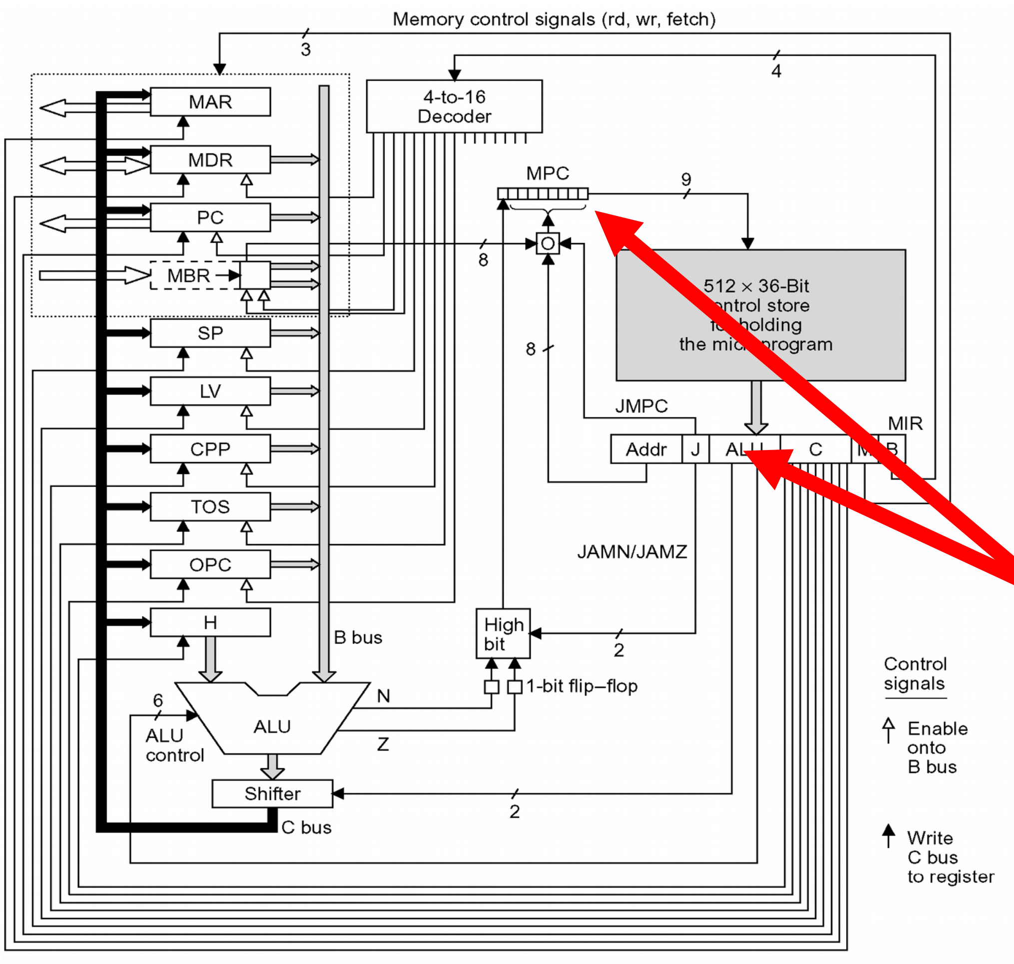


Figure 4-6. The complete block diagram of our example microarchitecture, the Mic-1.

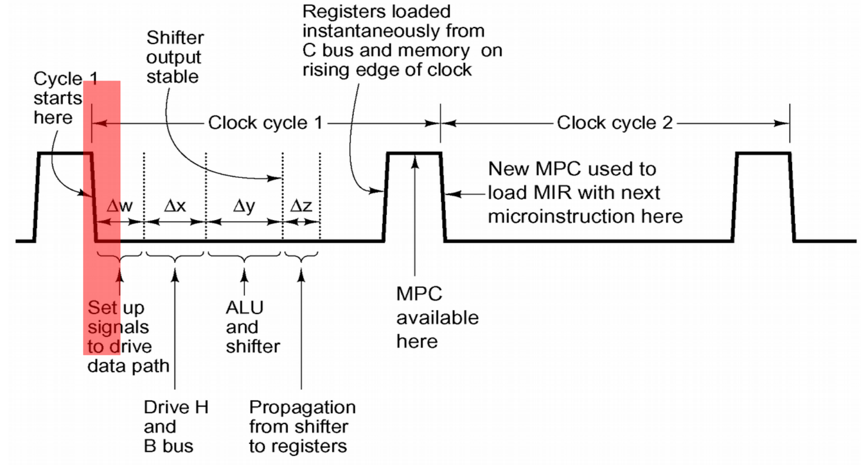


Figure 4-3. Timing diagram of one data path cycle.

- MIR is loaded on the falling edge of the clock based on the MPC address, control signals propagate

1. ALU Operation: N and Z values available and saved

Driving control signals

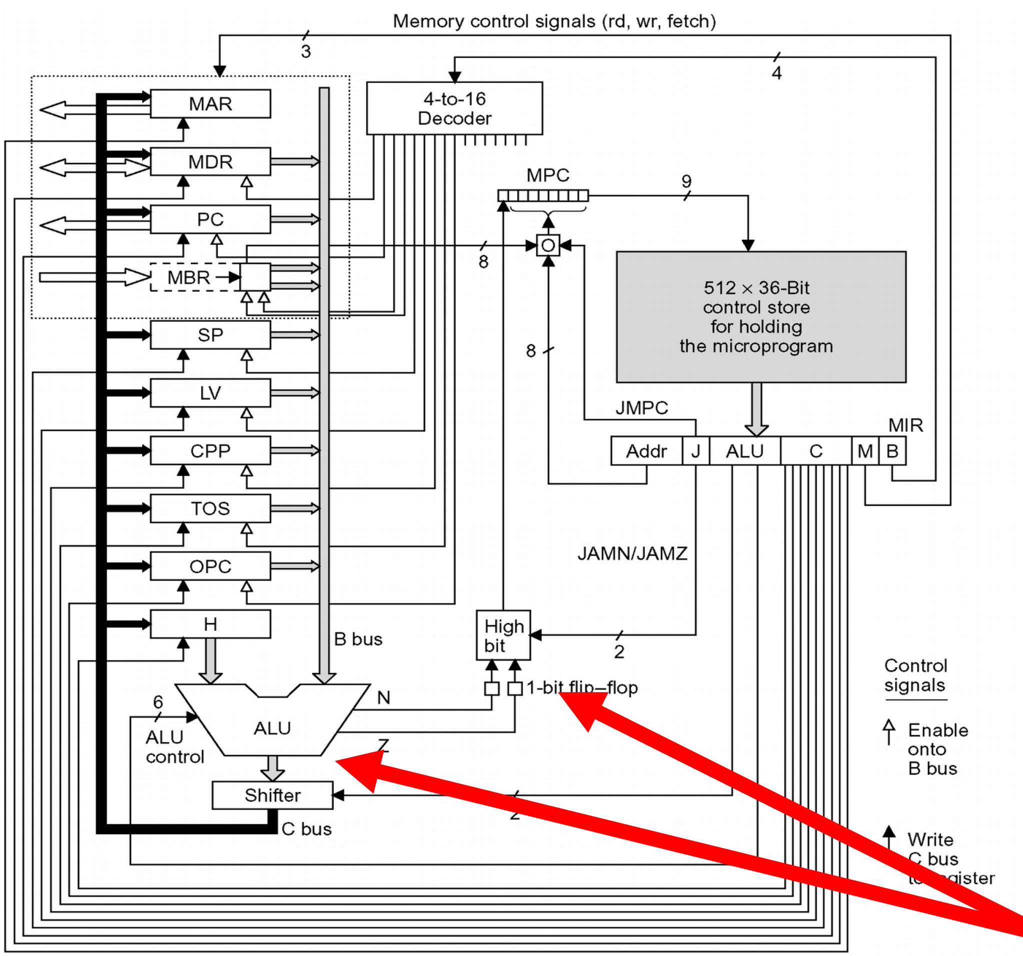


Figure 4-6. The complete block diagram of our example microarchitecture, the Mic-1.

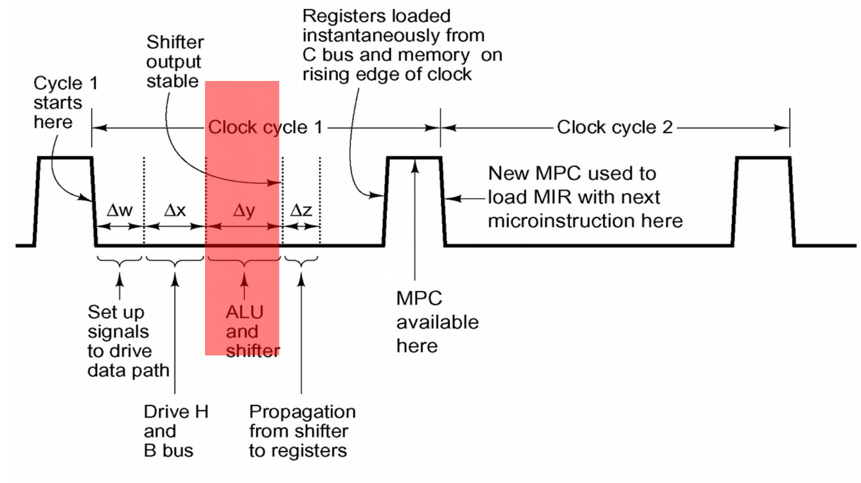
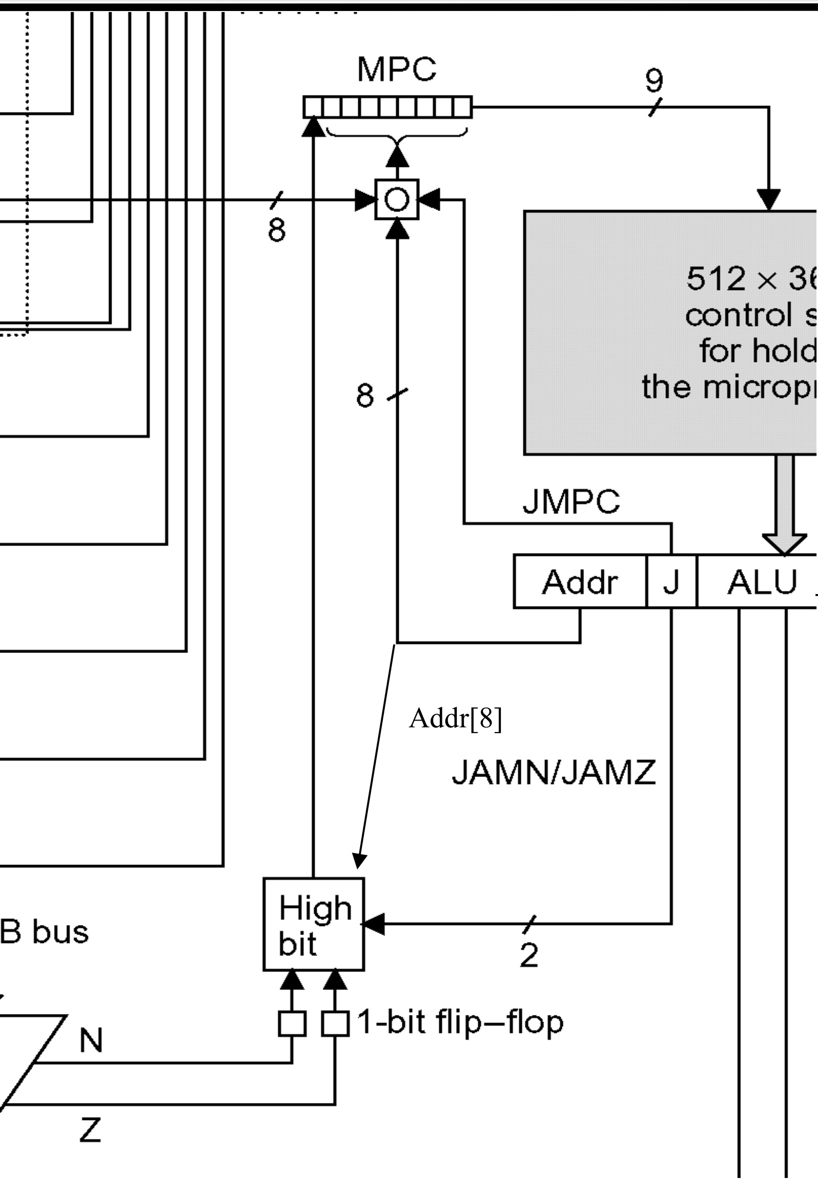


Figure 4-3. Timing diagram of one data path cycle.

- MIR is loaded on the falling edge of the clock based on the MPC address, control signals propagate

1. ALU Operation: N and Z values available and saved

Next microinstruction (1)

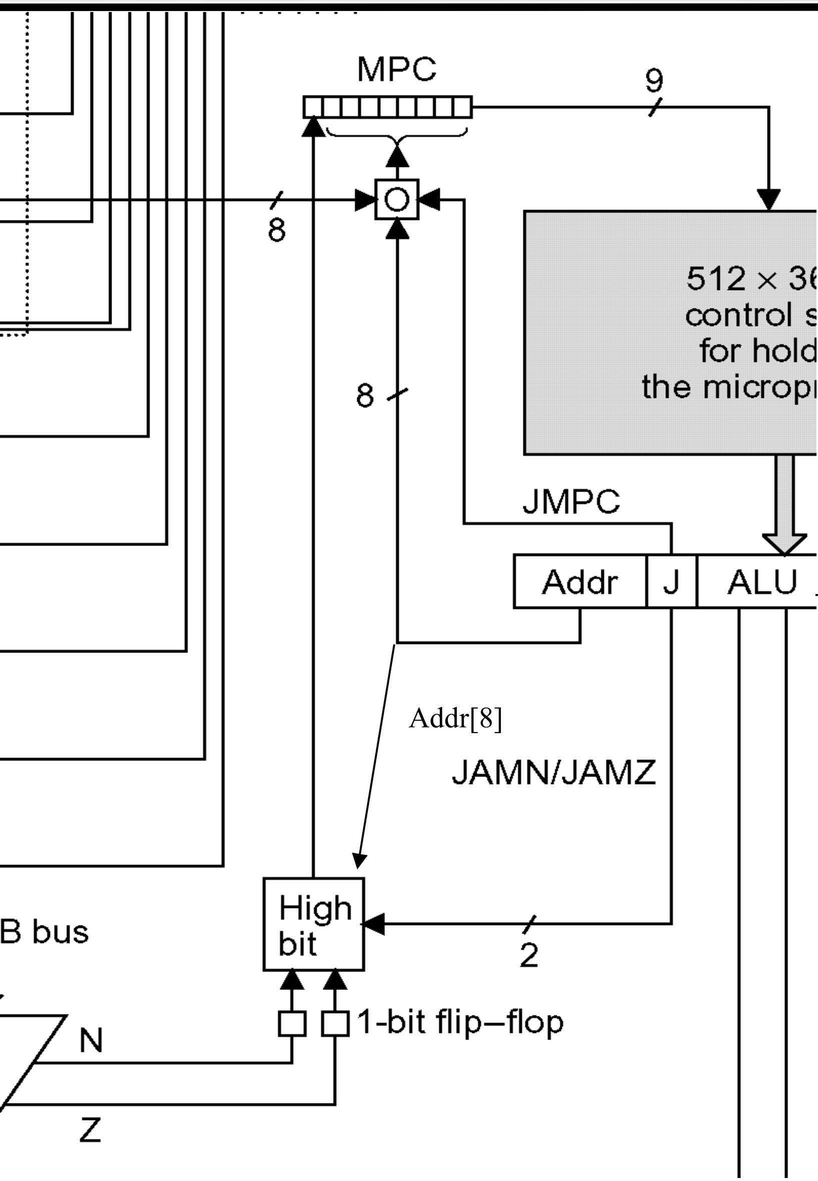


Addr (the address of the next microinstruction coded in the current microinstruction) **is copied in the MPC** (lower 8 bits, high bit is 0)

If **J is 000** the **next address is in the MPC** and the next microinstruction can be read from the control store (Note: microinstruction are not stored in the same order as Figure 4-17)

If **J is not 000** it is necessary to compute the **next microaddress depending on the values of J, N and Z** (whose value has been saved in flip-flop because the ALU returns correct result as long as data is passing through it)

Next microinstruction (2)



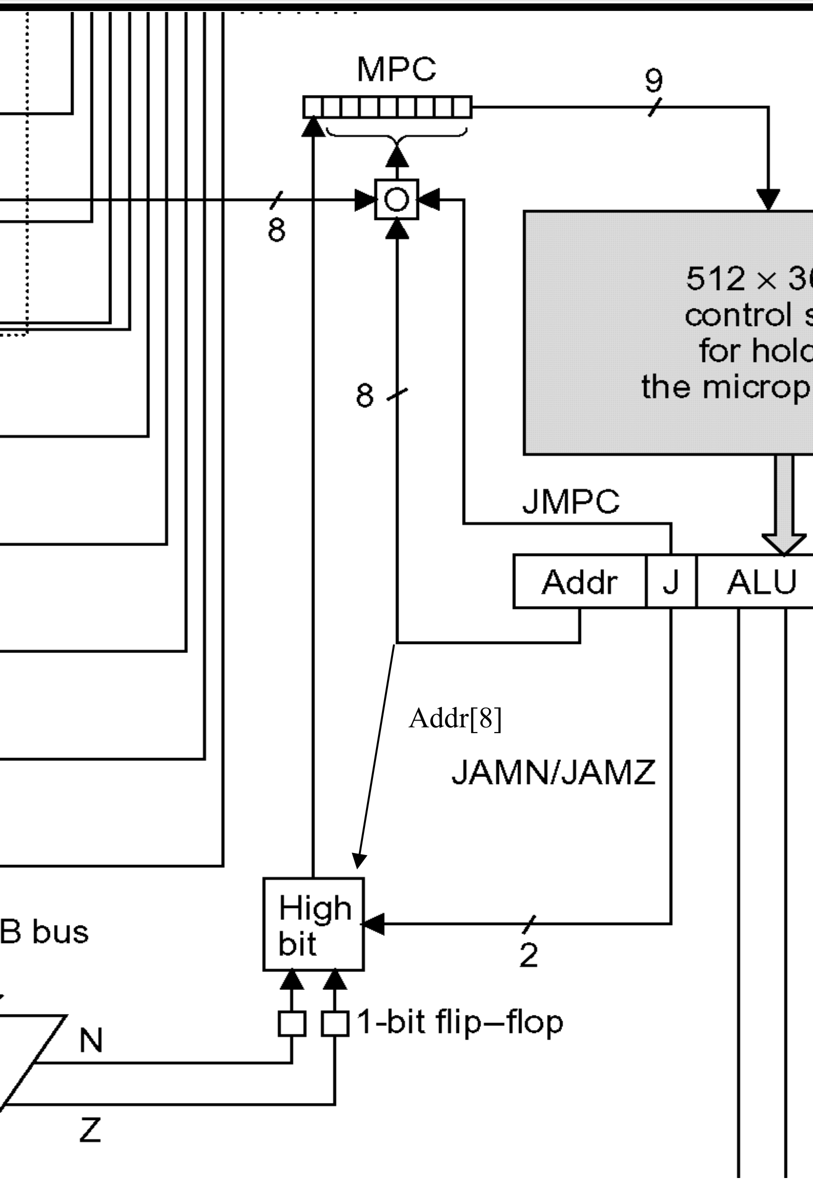
If JAMN or JAMZ are set to 1, the 'High bit' function computes the value of the high bit of the MPC as follows:

F = (JAMZ and Z) or (JAMN and N) or Addr[8]

(To avoid confusion: Addr[8] is in fact the 9th bit, the highest, of Addr, as bits count start from 0)

So the MPC can assume either the **value of Addr** or the value of **Addr with the high bit ORred with 1**

Next microinstruction (3)



F = (JAMZ and Z) or (JAMN and N) or Addr[8]

An example:

Let Addr <= 0xFF (or we would get the same value, 0xFF in either case)

Let JAMZ = 1 (or JAMN = 1)

Let Z=1 (or N=1)

in this case MPC is Addr + 0x100 (for example: if Addr=0x92, MPC = 0x92 + 0x100 = 0x192)

Note: 0x100 = 256

Microinstructions (4)

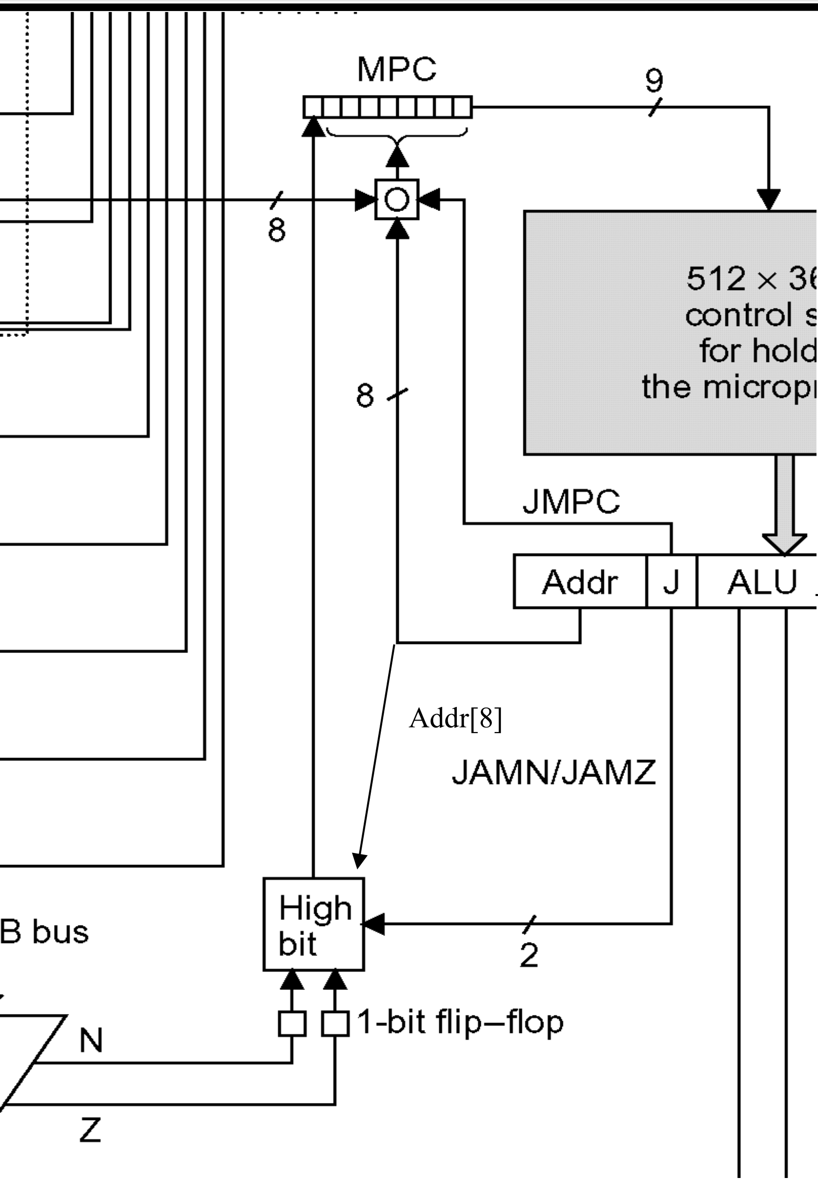
...but why is all that stuff required to determine the next microinstruction ?

Reason: **efficiency**

In case of conditional jumps (if..then..else) we normally need two jump addresses as parameter.

To uniform the microinstruction format we want all instruction to have the same length: either we make all microinstruction contain two addresses (-> waste of space) or (better solution) we specify only one address and compute the second one as Addr + *Constant Value* (in Mic-1 *Constant Value* = 0x100)

Next microinstruction (5)



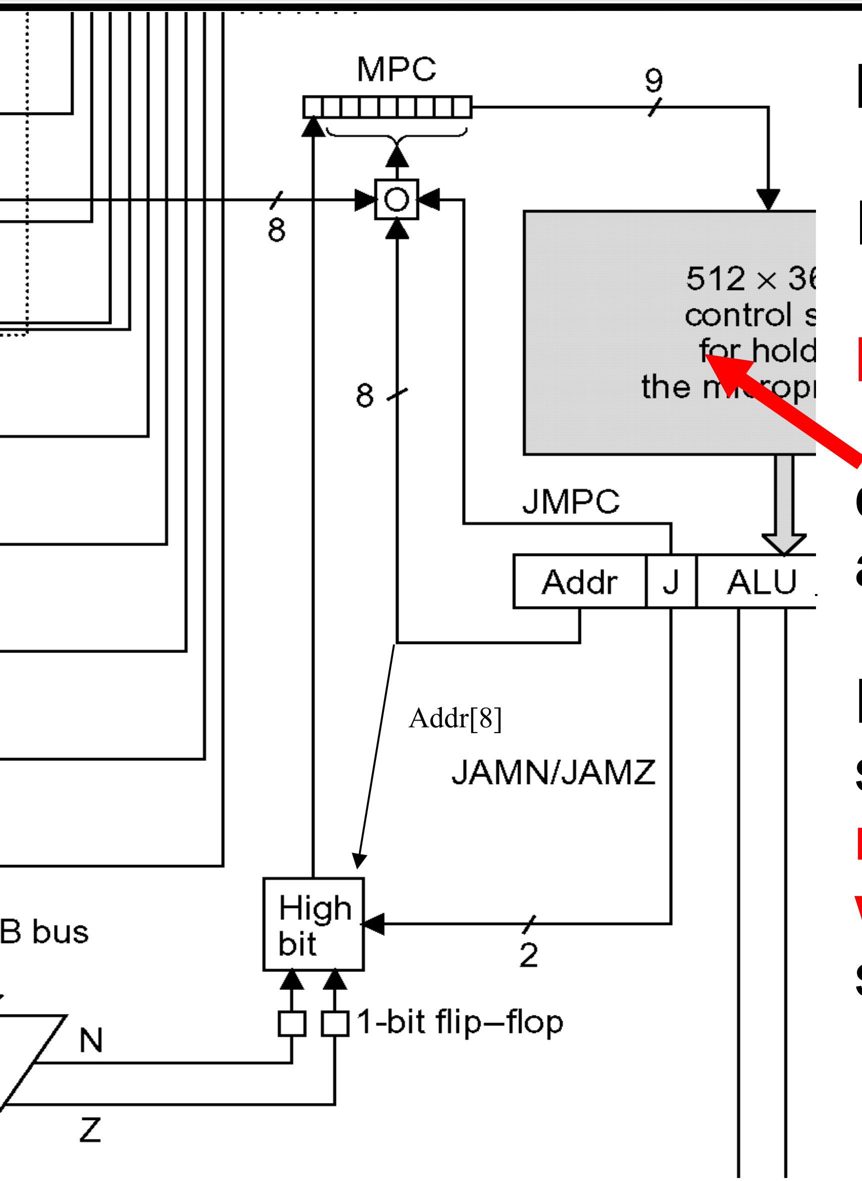
If JMPC = 0, Addr is copied to MPC

If JMPC = 1, **the lower 8-bits of Addr are ORred with the MBR value**, and the result is put in the MPC

Normally when JMPC = 1, Addr is set to either 0x000 or 0x100

JMPC is used to jump to the address specified by the MBR, which, as we will see, contains the opcode of the ISA instruction: in fact, **microinstruction for each macroinstruction are stored starting from the position determined by the opcode of the latter.**

Next microinstruction (6)



Example

ISA instruction:

BIPUSH **opcode is 0x10**

corresponding microinstructions starts at address 0x10 in the control store

For the reasons explained in the previous slides, it is clear that the **next microinstruction can be determined only when the MBR, N and Z are ready**, i.e. starting from the successive clock pulse)