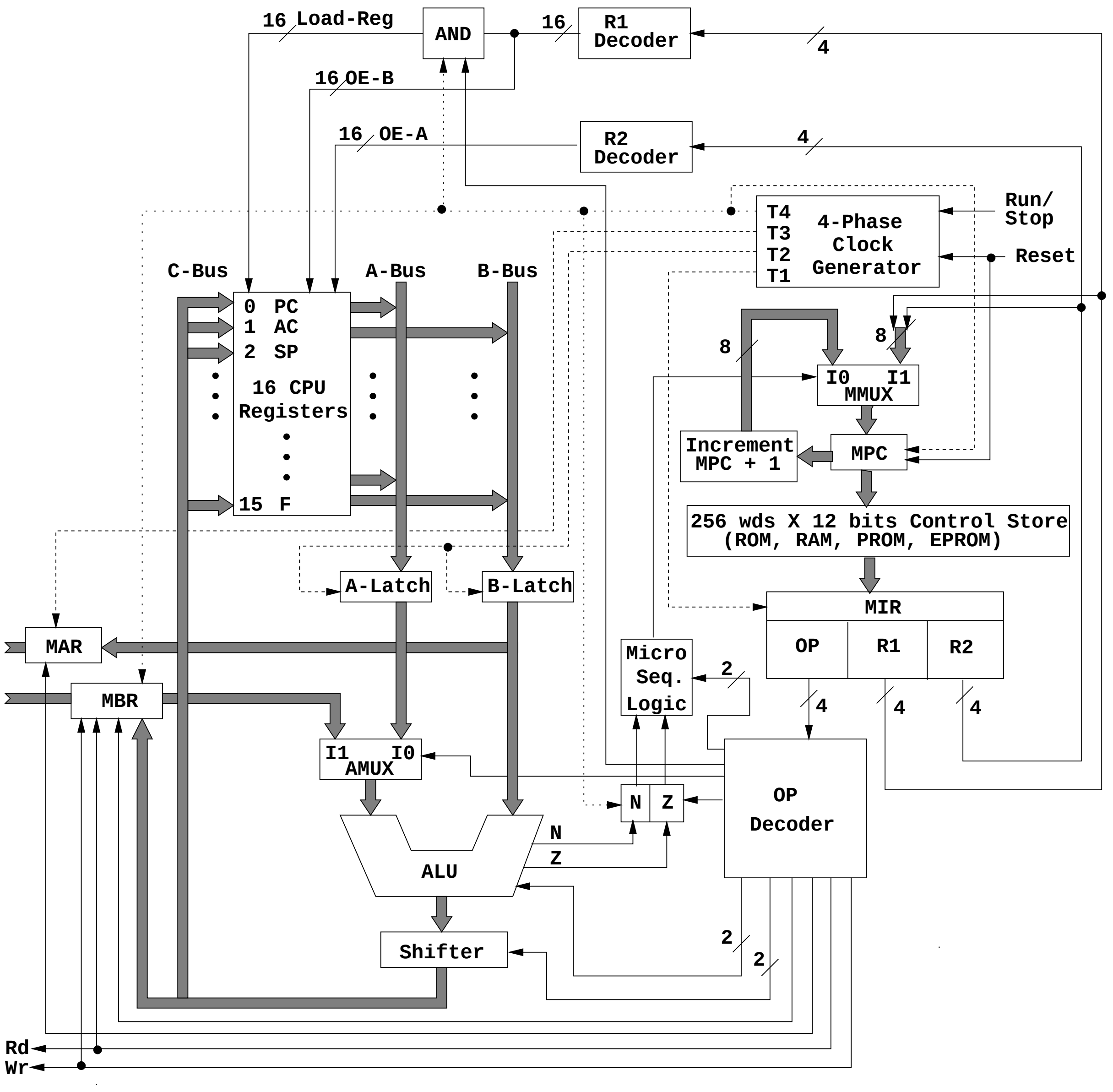

Figure 1: Block diagram for vertically microprogrammed microarchitecture (Mic-2)

MIC-2 Control Signals														
OpCode Decimal	Mnemonic	ALU F1	ALU F0	SHFT S1	SHFT S0	Latch NZ	AMUX	ENC AND	MAR	MBR	RD	WR	COND C1	COND C0
0	ADD					+		+						
1	AND		+			+		+						
2	MOVE	+				+		+						
3	COMPL	+	+			+		+						
4	LSHIFT	+		+		+		+						
5	RSHIFT	+			+	+		+						
6	GETMBR	+				+	+	+						
7	TEST	+				+								
8	BEGRD	+							+		+			
9	BEGWR	+							+	+		+		
10	CONRD	+									+			
11	CONWR	+										+		
12														
13	NJUMP	+												+
14	ZJUMP	+											+	
15	UJUMP	+											+	+

Control signals generated by the opcode decoder for each mic2 microinstruction opcode. Note: A plus means the signal is asserted (i.e., set equal to 1); a blank means it is negated (i.e., set equal to 0). The two clocked D-latches used to remember the N and Z signals coming from the ALU are controlled by the "Latch NZ" signal, which is ANDed with timing signal T4; so that if the Latch NZ signal is asserted (i.e., equal to 1) and it is timing phase T4, then the clock line into the N and Z latches goes to logic 1 and these latches copy and hold the ALU's N and Z signal values, respectively, for later use by either a ZJUMP or NJUMP microinstruction. The Latch NZ control signal generated by the opcode decoder and the addition of the N and Z latches are the main differences between the mic2 and the mic1; the remaining 12 control signals generated by the mic2's opcode decoder are the same as those used to control the mic1's data path and, thus, perform the same functions as those specified in the Microinstruction format (32-bit word) for the mic1.

Figure 2: Mic-2 opcode decoding and control signals

MIC-2 OpCodes			
OpCode Binary	**Mnemonic & Operands**	**Instruction**	**Meaning or Action**
0000	`ADD` r1,r2	Addition	r1:= r1+r2
0001	`AND` r1,r2	Boolean AND	r1:= r1.AND.r2 = band(r1,r2)
0010	`MOVE` r1,r2	Move register	r1:= r2
0011	`COMPL` r1,r2	Complement	r1:= inv(r2)
0100	`LSHIFT` r1,r2	Left shift	r1:= lshift(r2)
0101	`RSHIFT` r1,r2	Right shift	r1:= rshift(r2)
0110	`GETMBR` r1	Store MBR in register	r1:= mbr
0111	`TEST` r2	Test register	**if** r2<0 **then** N:=1; **if** r2=0 **then** Z:=1
1000	`BEGRD` r1	Begin read	mar:= r1; rd
1001	`BEGWR` r1,r2	Begin write	mar:= r1; mbr:=r2; wr
1010	`CONRD`	Continue read	rd
1011	`CONWR`	Continue write	wr
1100		(not used)	
1101	`NJUMP` r	Jump if N=1	**if** n **then go to** r
1110	`ZJUMP` r	Jump if Z=1	**if** z **then go to** r
1111	`UJUMP` r	Unconditional jump	**go to** r

Note: r = 16*r1 + r2; i.e., i.e., r is the 8-bit concatenation [r1r2] of the two 4-bit fields specified by r1 and r2 in the left to right order r1 followed by r2. In translating this assembly code decimal value r is converted to an 8-bit binary value ($0 \leq r \leq 255$) and the high order 4 bits are placed in the r1 field and the low order 4 bits are placed in the r2 field. Also, r1 and r2 are each a 4-bit designator for one of the 16 CPU registers in the scratchpad, and both could, if desired, specify the same register in a valid mic2 instruction. Furthermore, the notation "if r1 < 0" means that "if the contents of the register named in the r1 field is less than zero" then do something. In this case N and Z refer to D-latches that save the combinational values coming out of the ALU on the n and z output wires if enabled to do so by NZ control signal.

Figure 3: Table of Mic-2 (micro) Instructions