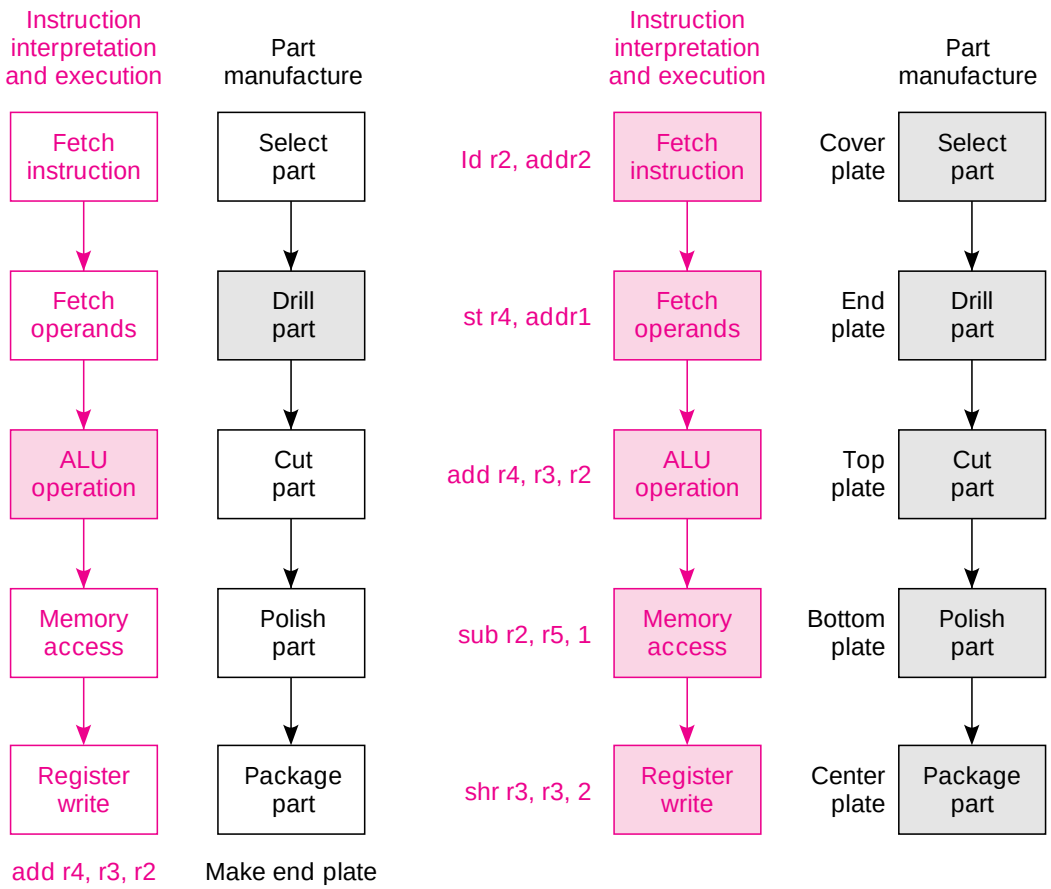
Fig 5.1 Executing Machine Instructions versus Manufacturing Small Parts

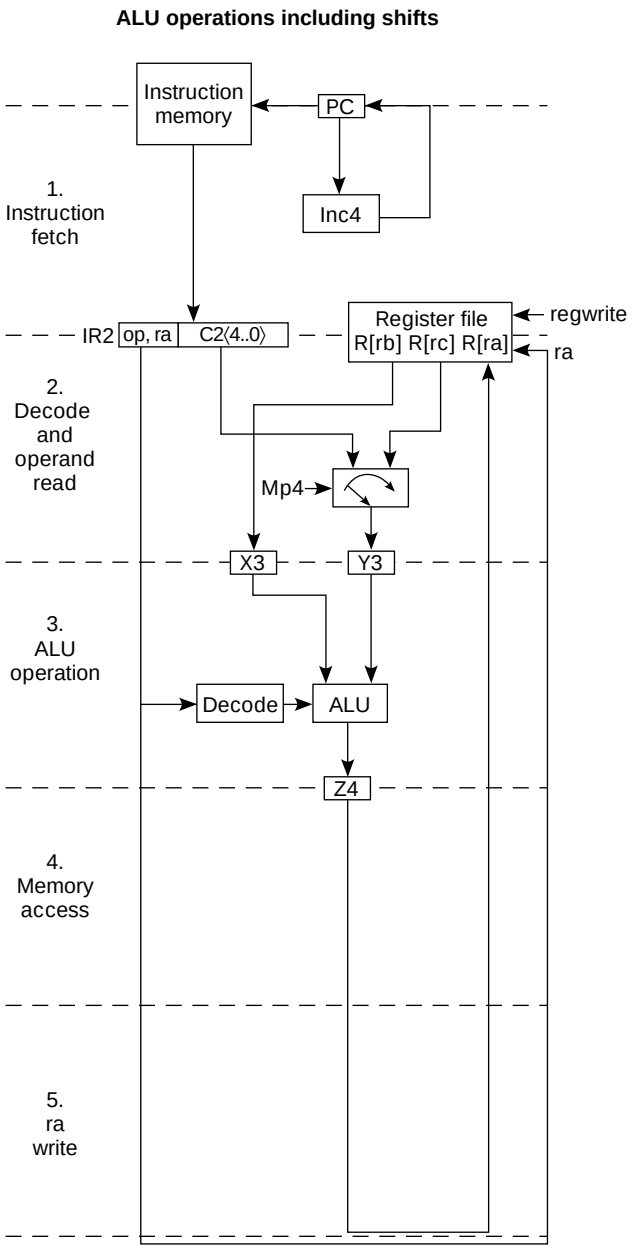


(a) Without pipelining/assembly line **(b) With pipelining/assembly line**

© 1997 V. Heuring and H. Jordan

Fig 5.2 ALU Instructions

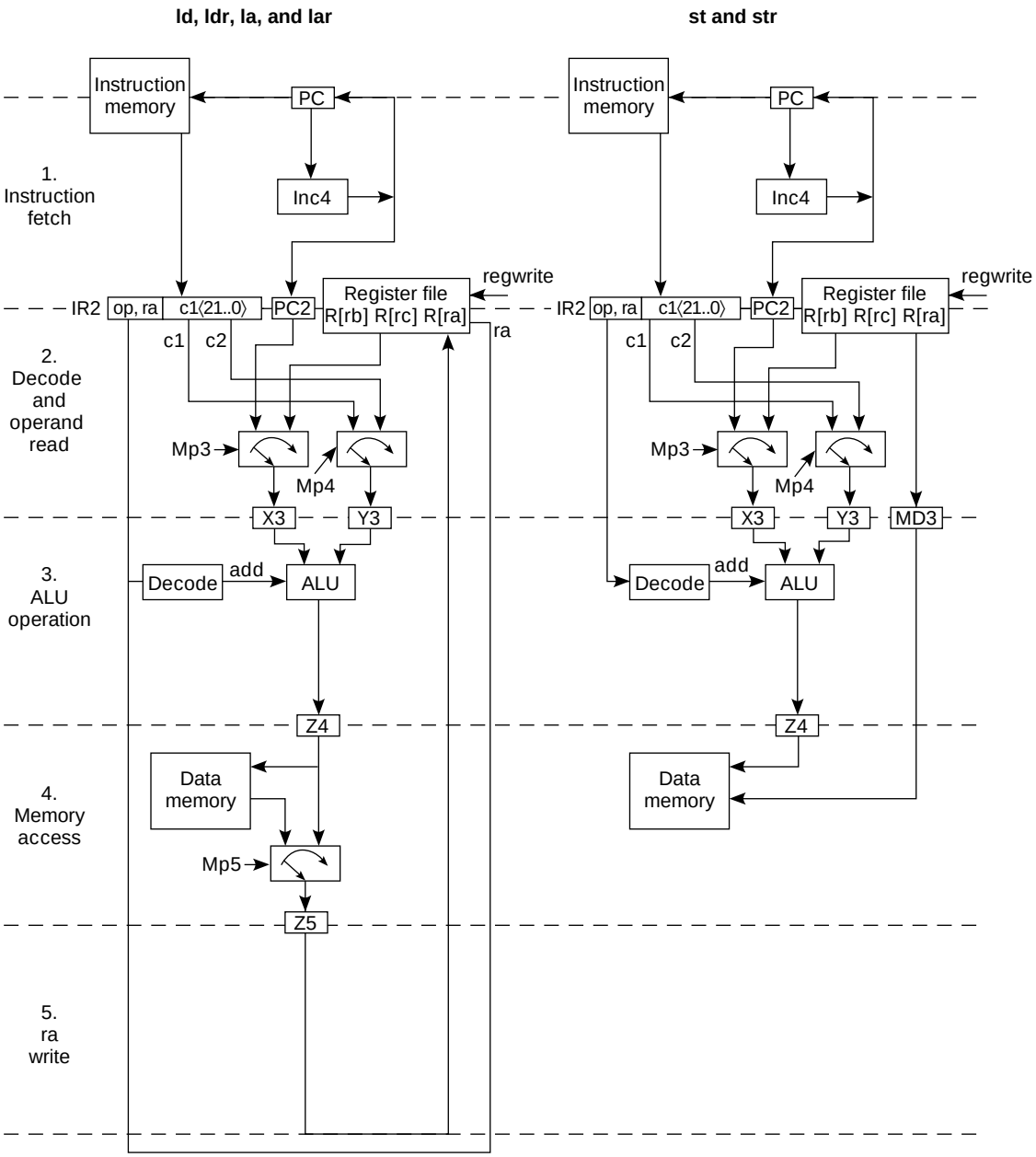
- Instructions fit into 5 stages
- Second ALU operand comes either from a register or instruction register c2 field
- Opcode must be available in stage 3 to tell ALU what to do
- Result register, ra, is written in stage 5
- No memory operation



© 1997 V. Heuring and H. Jordan

Fig 5.4 The Memory Access Instructions: ld, ldr, st, and str

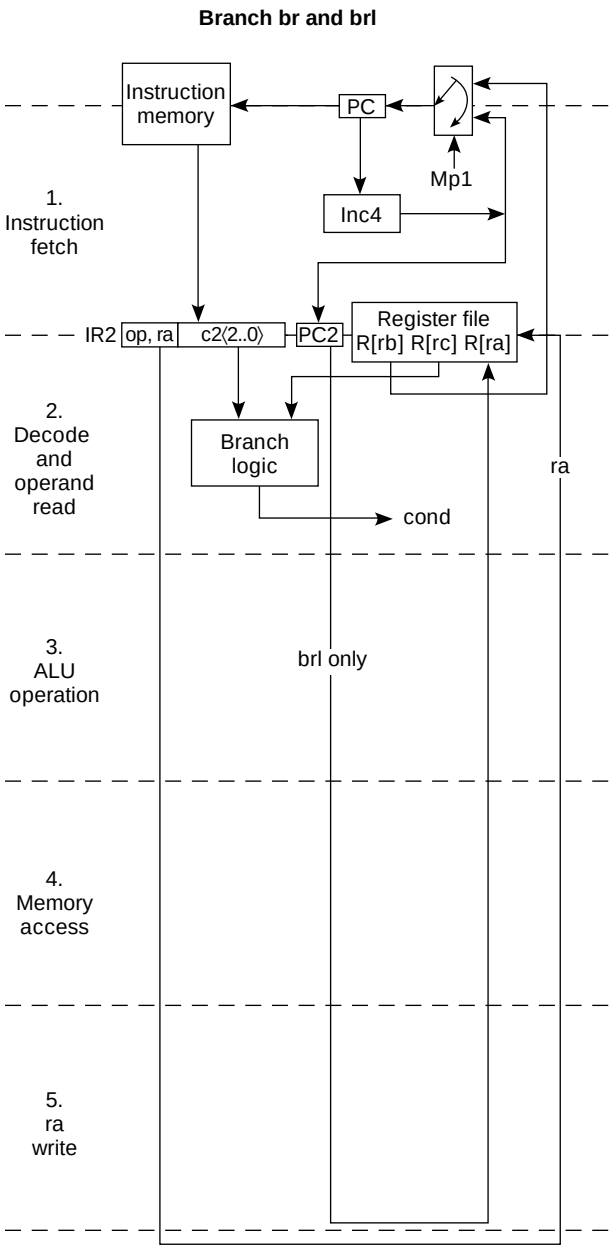
- ALU computes effective addresses
- Stage 4 does read or write
- Result register written only on load



© 1997 V. Heuring and H. Jordan

Fig 5.5 The Branch Instructions

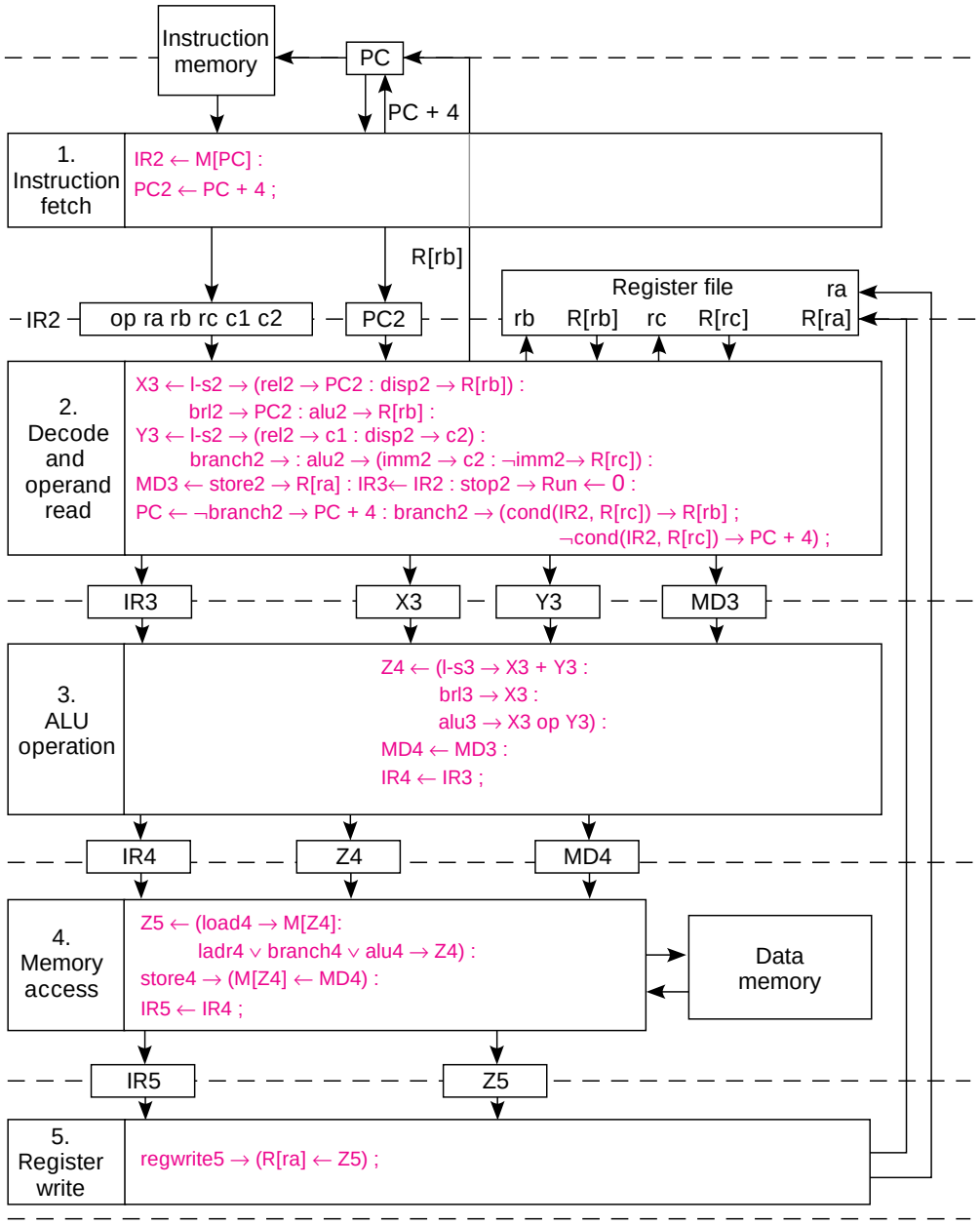
- The new program counter value is known in stage 2—but not in stage 1
- Only branch and link does a register write in stage 5
- There is no ALU or memory operation



© 1997 V. Heuring and H. Jordan

Fig 5.6 The SRC Pipeline Registers and RTN Specification

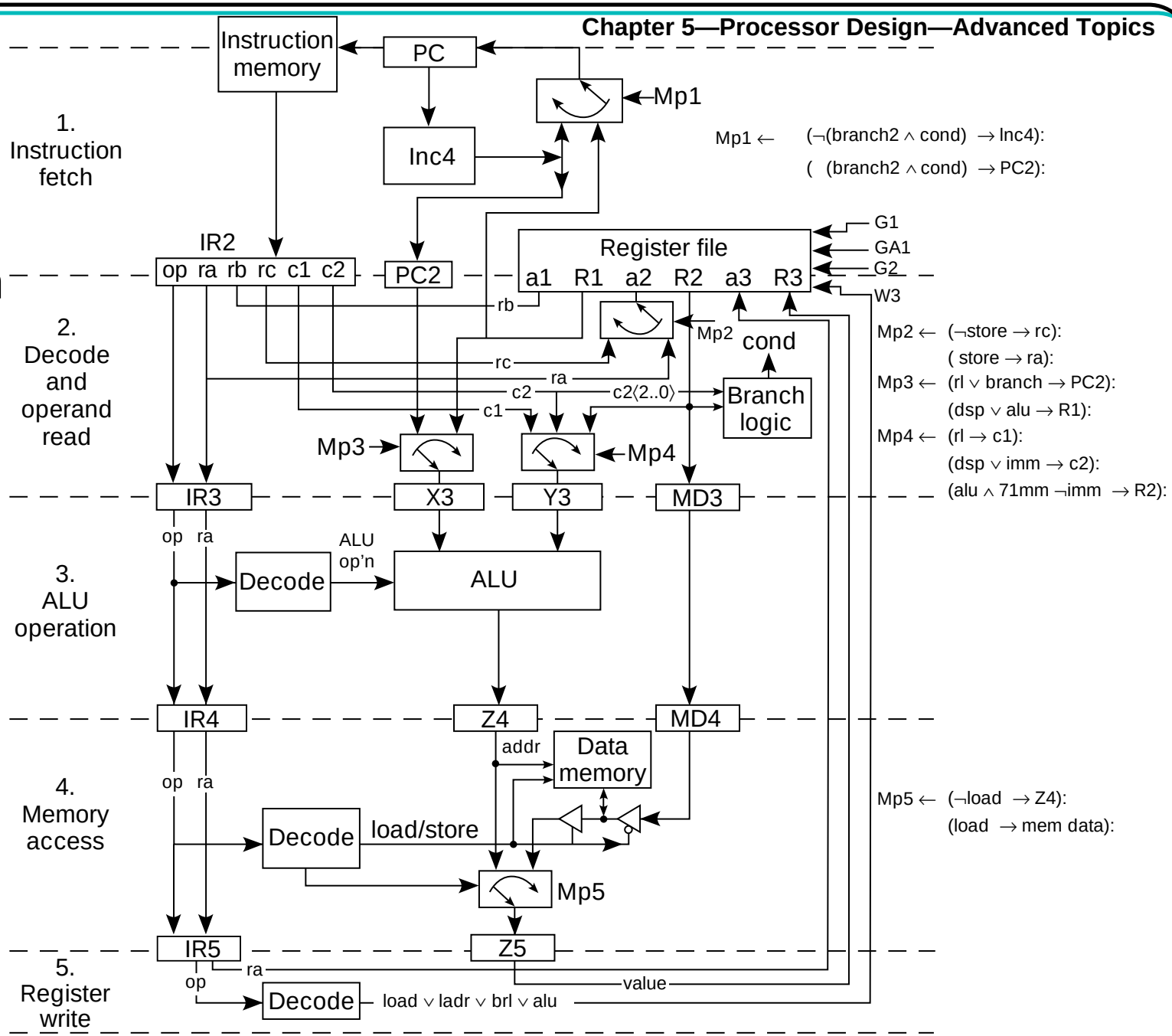
- The pipeline registers pass information from stage to stage
- RTN specifies output register values in terms of input register values for stage
- Discuss RTN at each stage on blackboard



© 1997 V. Heuring and H. Jordan

Fig 5.7 The Pipeline Data Path with Selected Control Signals

- Most control signals shown and given values
- Multi-plexer control is stressed in this figure

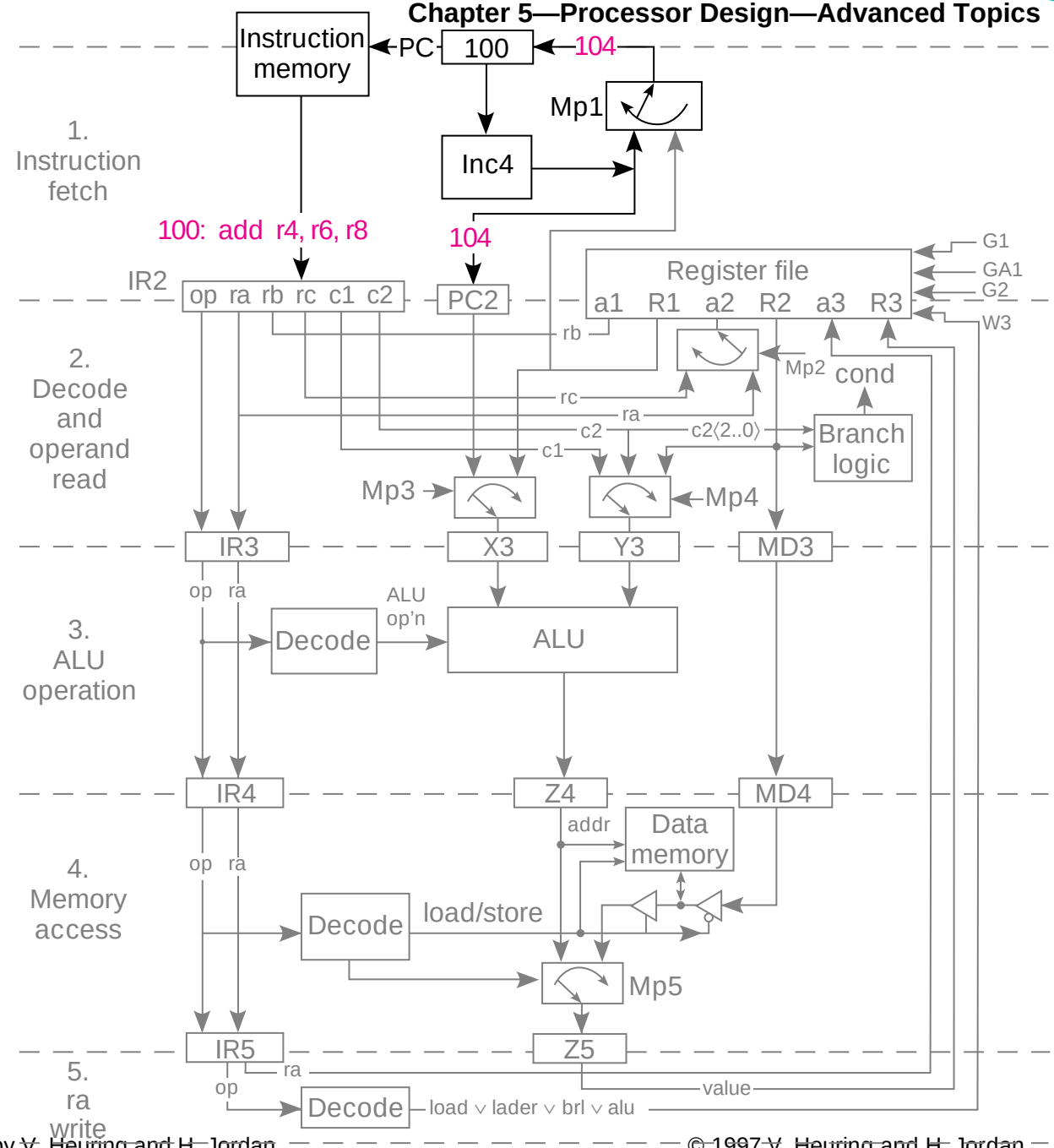


© 1997 V. Heuring and H. Jordan

Fig 5.8 First Clock Cycle: add Enters Stage 1 of Pipeline

- **Program counter is incremented to 104**

512: sub ...
. . . . . .
112: str r12, #32
108: brl r9, r11, 001
104: ld r7, r5, #128
100: add r4, r6, r8

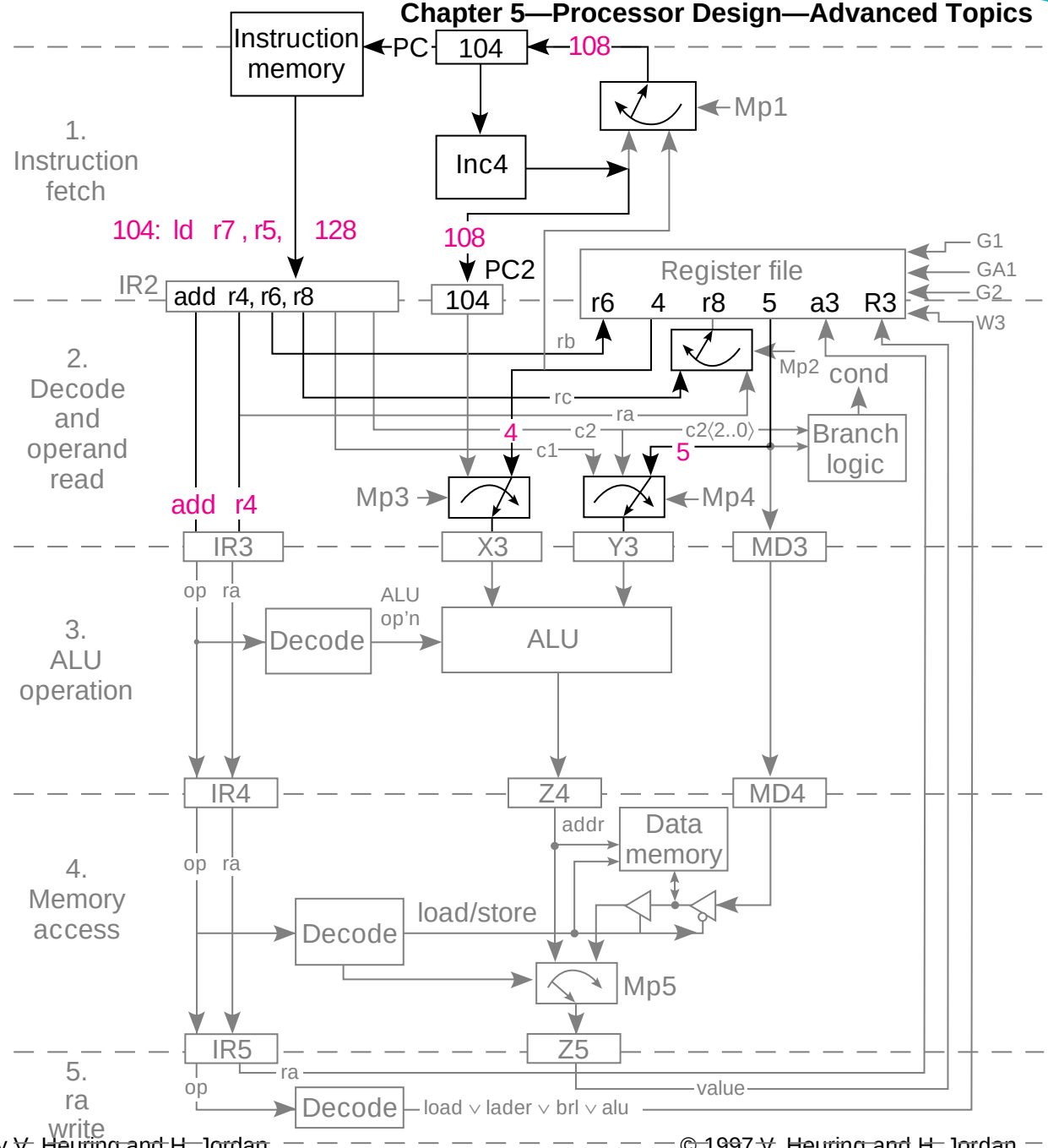


© 1997 V. Heuring and H. Jordan

Fig 5.9 Second Clock Cycle: add Enters Stage 2, While 1d is Being Fetched at Stage 1

- add operands are fetched in stage 2

512: sub ...
. . . . . .
112: str r12, #32
108: brl r9, r11, 001
104: ld r7, r5, #128
100: add r4, r6, r8

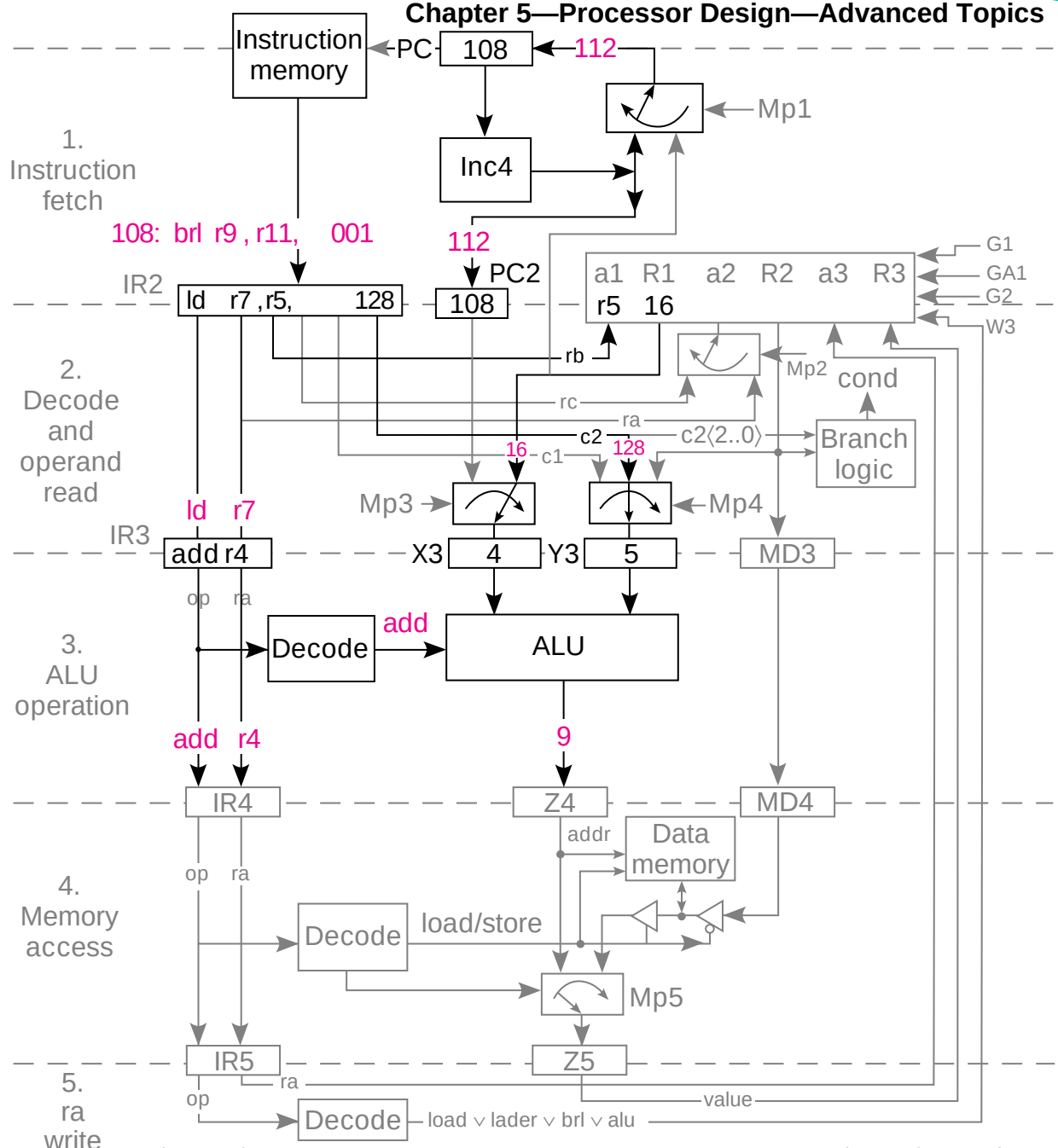


© 1997 V. Heuring and H. Jordan

Fig 5.10 Third Clock Cycle: `brl` Enters the Pipeline

- **add performs its arithmetic in stage 3**

512: sub ...
. . . . . .
112: str r12, #32
108: brl r9, r11, 001
104: ld r7, r5, #128
100: add r4, r6, r8

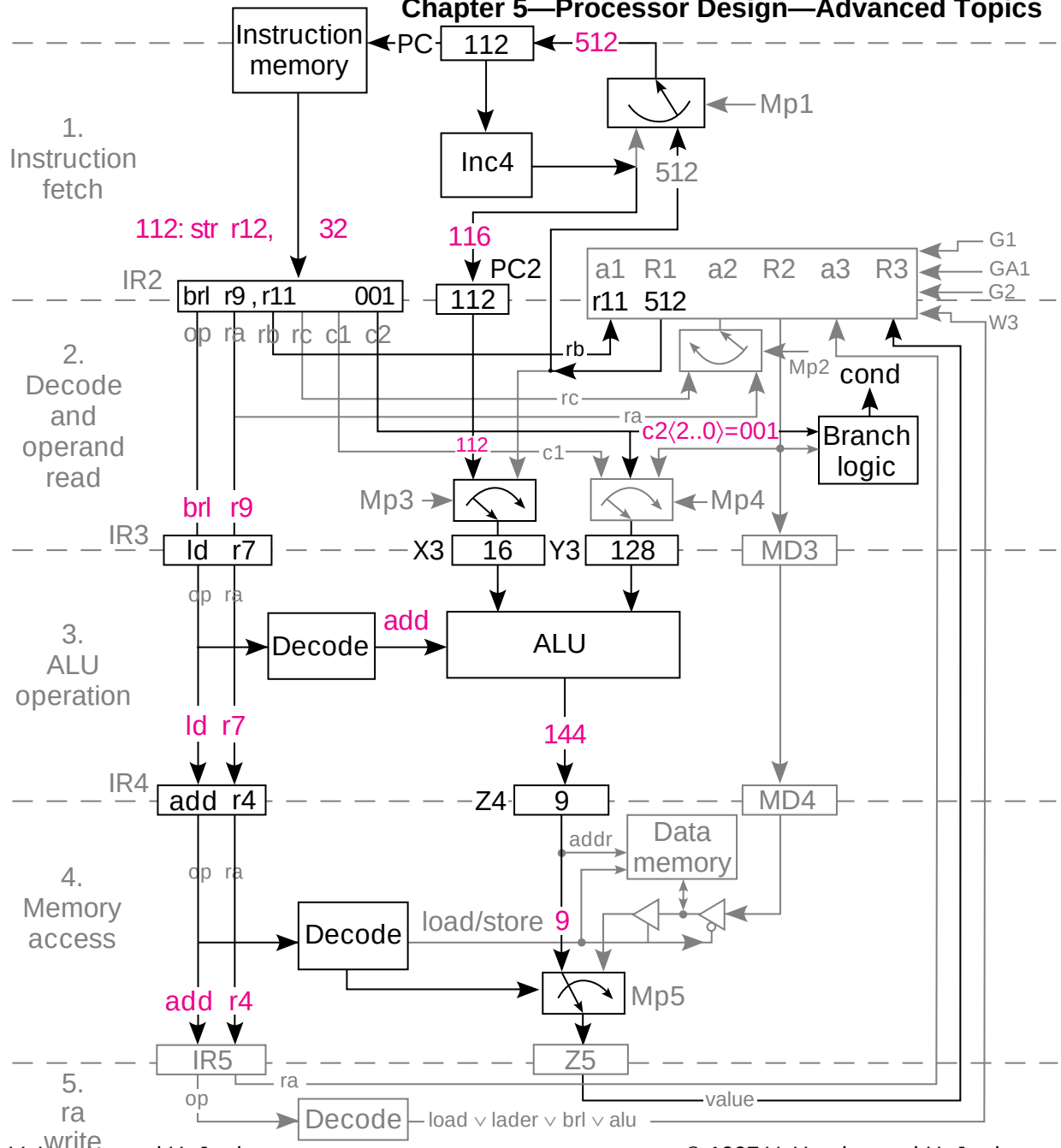


© 1997 V. Heuring and H. Jordan

Fig 5.11 Fourth Clock Cycle: `str` Enters the Pipeline

- add is idle in stage 4
- Success of `brl` changes program counter to 512

512: sub ...
. . . . . .
112: str r12, #32
108: brl r9, r11, 001
104: ld r7, r5, #128
100: add r4, r6, r8



© 1997 V. Heuring and H. Jordan

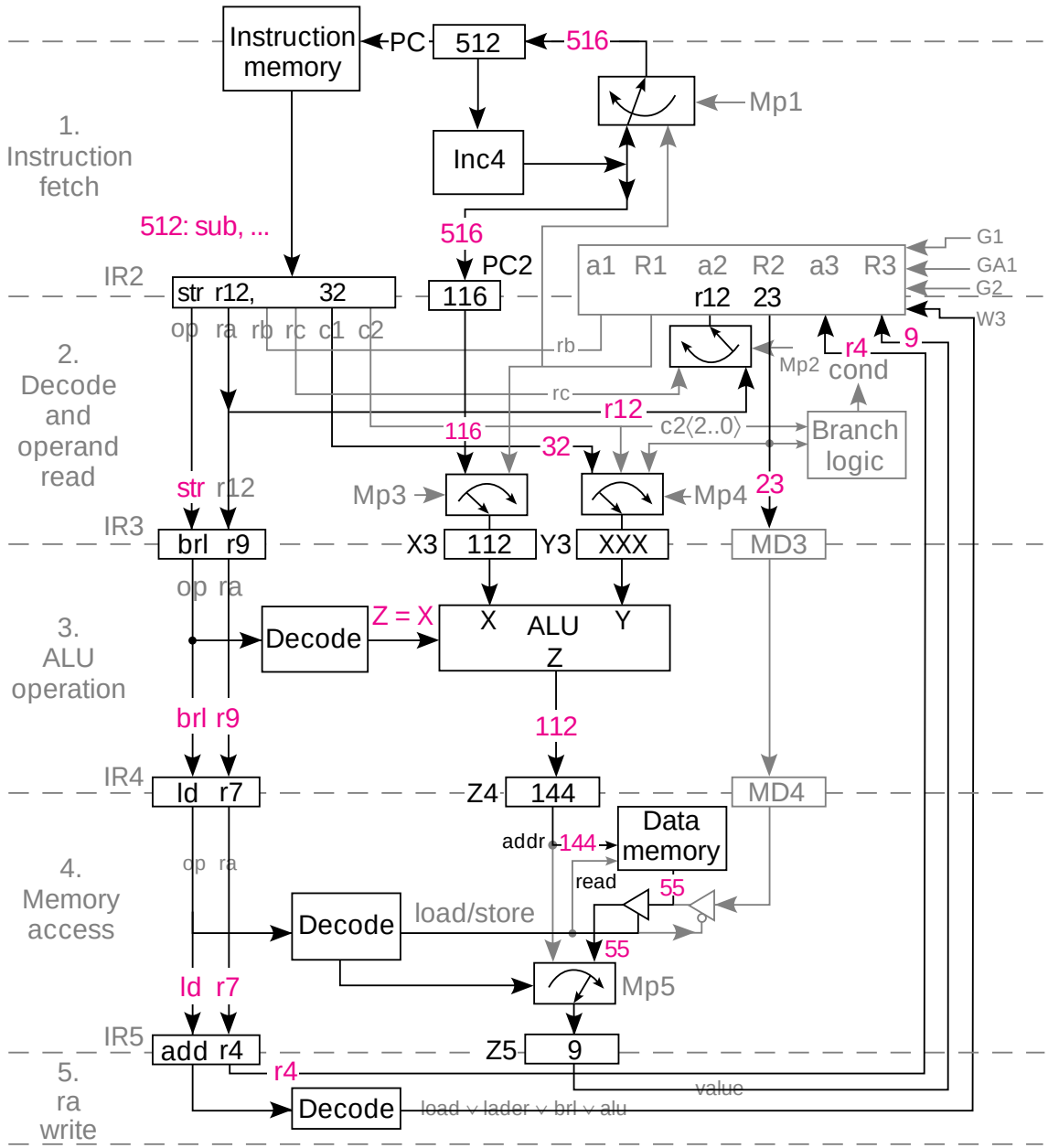
Fig 5.12 Fifth Clock Cycle: add Completes, sub Enters the Pipeline

- add completes in stage 5
- sub is fetched from location 512 after successful brl

512: sub ...

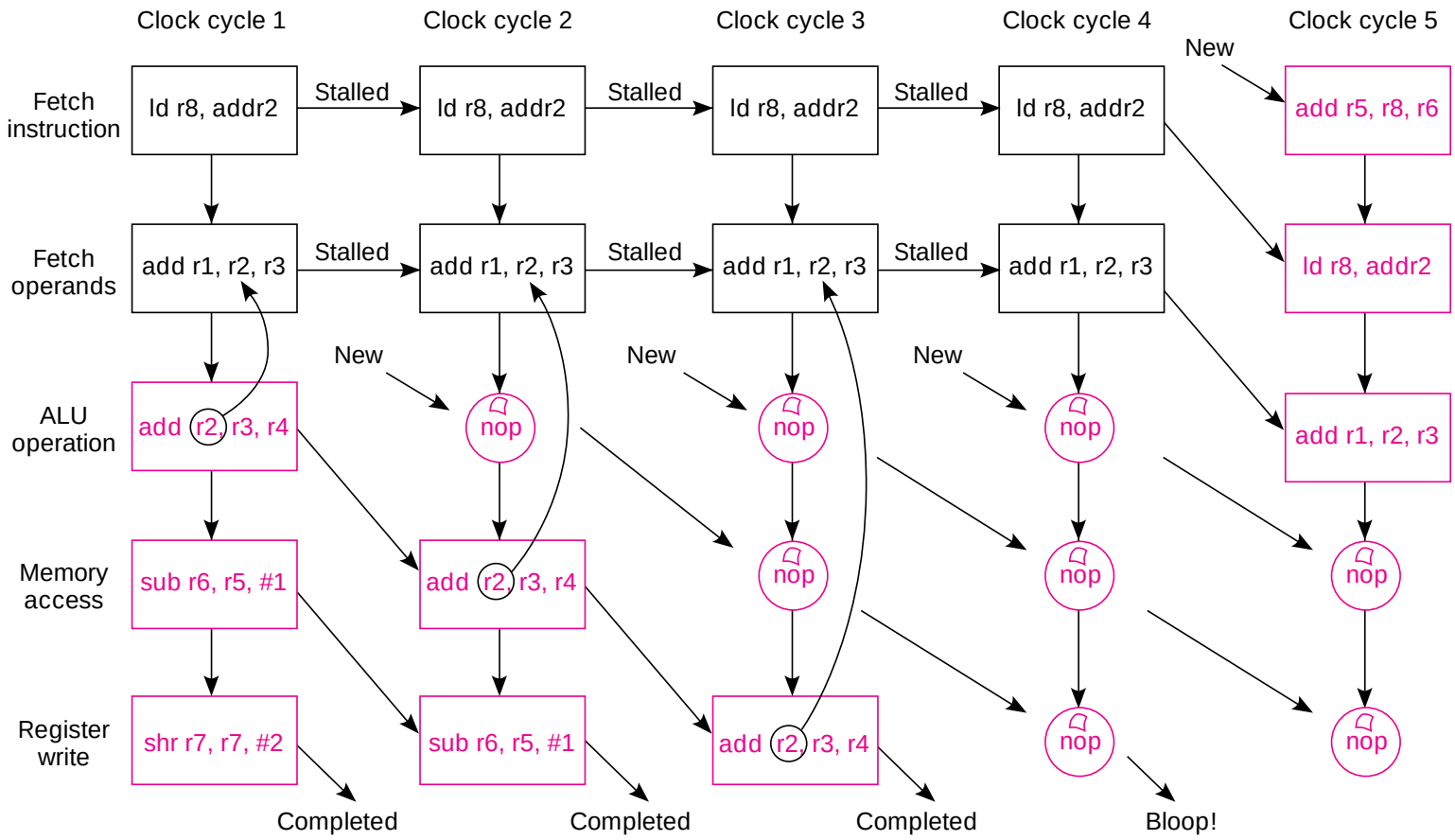
. . . . . .

112: str r12, #32
108: brl r9, r11, 001
104: ld r7, r5, #128
100: add r4, r6, r8



© 1997 V. Heuring and H. Jordan

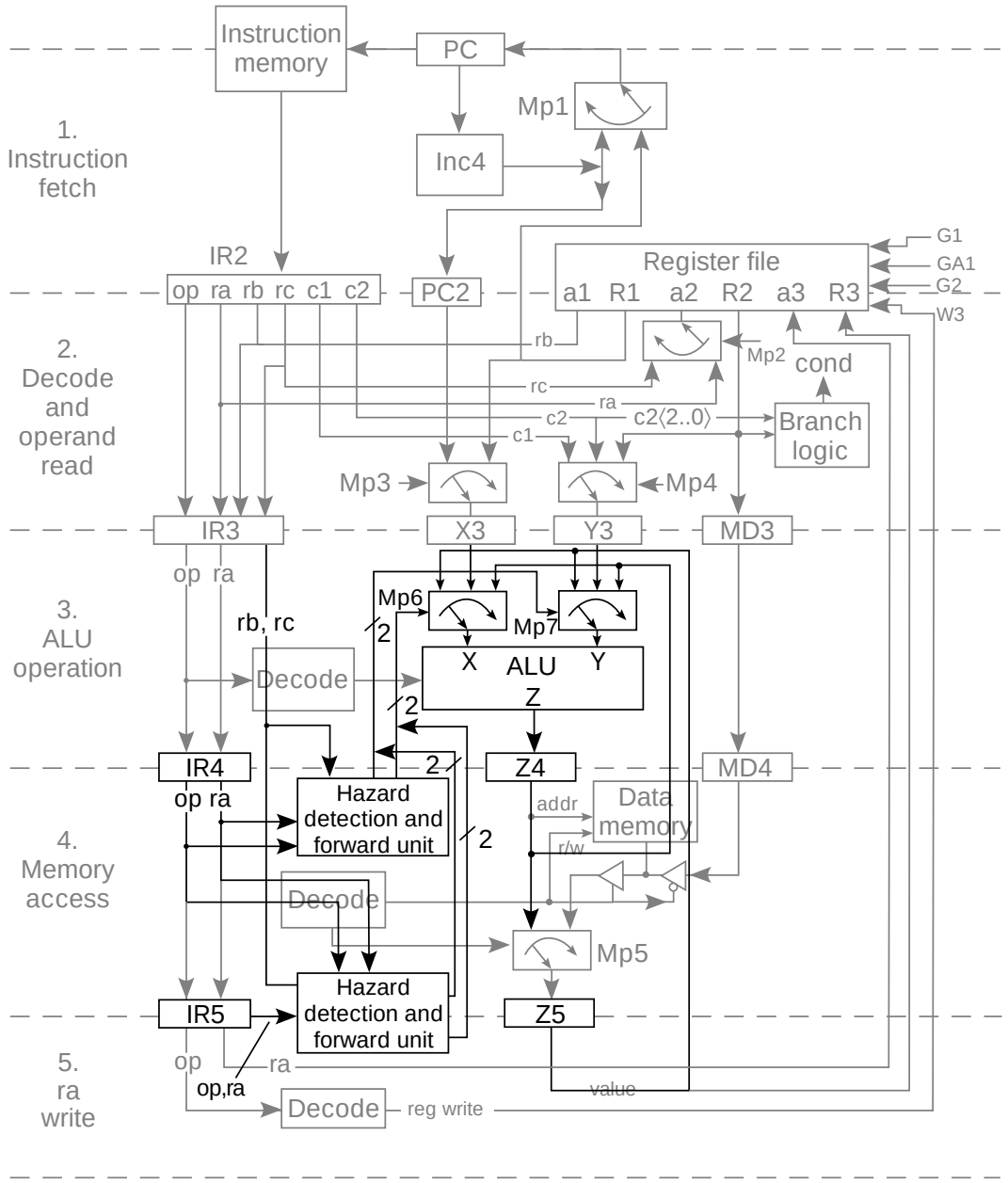
Fig 5.14 Stall Due to a Data Dependence Between Two ALU Instructions



© 1997 V. Heuring and H. Jordan

Fig 5.15 Hazard Detection and Forwarding

- Can be from either Z4 or Z5 to either X or Y input to ALU
- rb and rc needed in stage 3 for detection



©1996 Vincent P. Heuring and Harry F. Jordan
© 1997 V. Heuring and H. Jordan

Microprogramming: Basic Idea

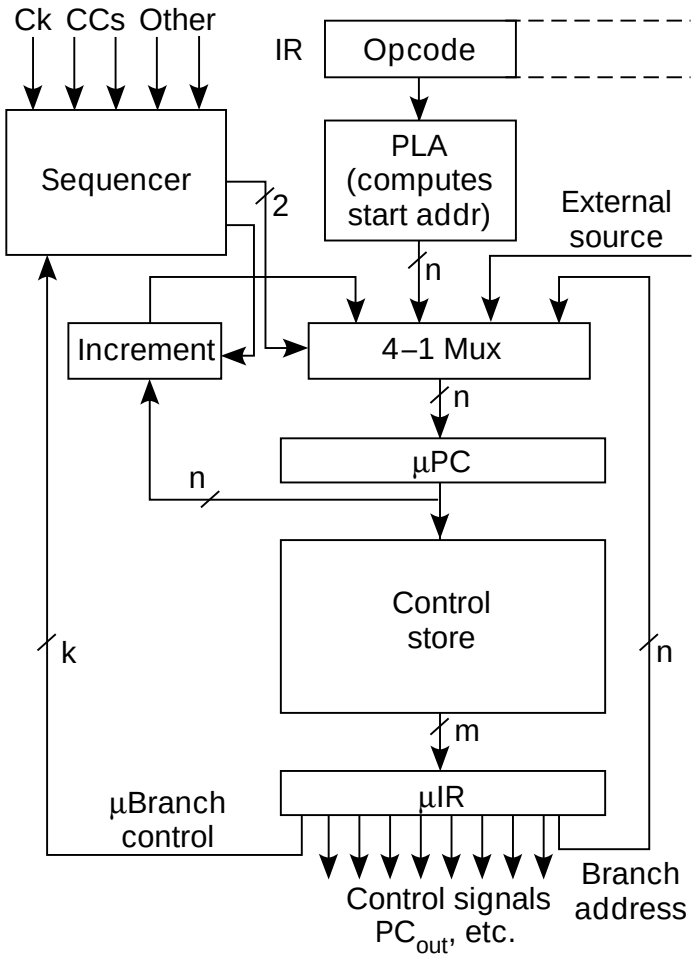
- Recall control sequence for 1-bus SRC

Step	Concrete RTN	Control Sequence
T0	MA ← PC: C ← PC + 4;	PC_{out}, MA_{in}, INC4, C_{in}, Read
T1	MD ← M[MA]: PC ← C;	C_{out}, PC_{in}, Wait
T2	IR ← MD;	MD_{out}, IR_{in}
T3	A ← R[rb];	Grb, R_{out}, A_{in}
T4	C ← A + R[rc];	Grc, R_{out}, ADD, C_{in}
T5	R[ra] ← C;	C_{out}, Gra, R_{in}, End

- Control unit job is to generate the sequence of control signals
- How about building a computer to do this?

© 1997 V. Heuring and H. Jordan

Fig 5.16 Block Diagram of Microcoded Control Unit



- **Microinstruction has branch control, branch address, and control signal fields**
- **Microprogram counter can be set from several sources to do the required sequencing**

© 1997 V. Heuring and H. Jordan

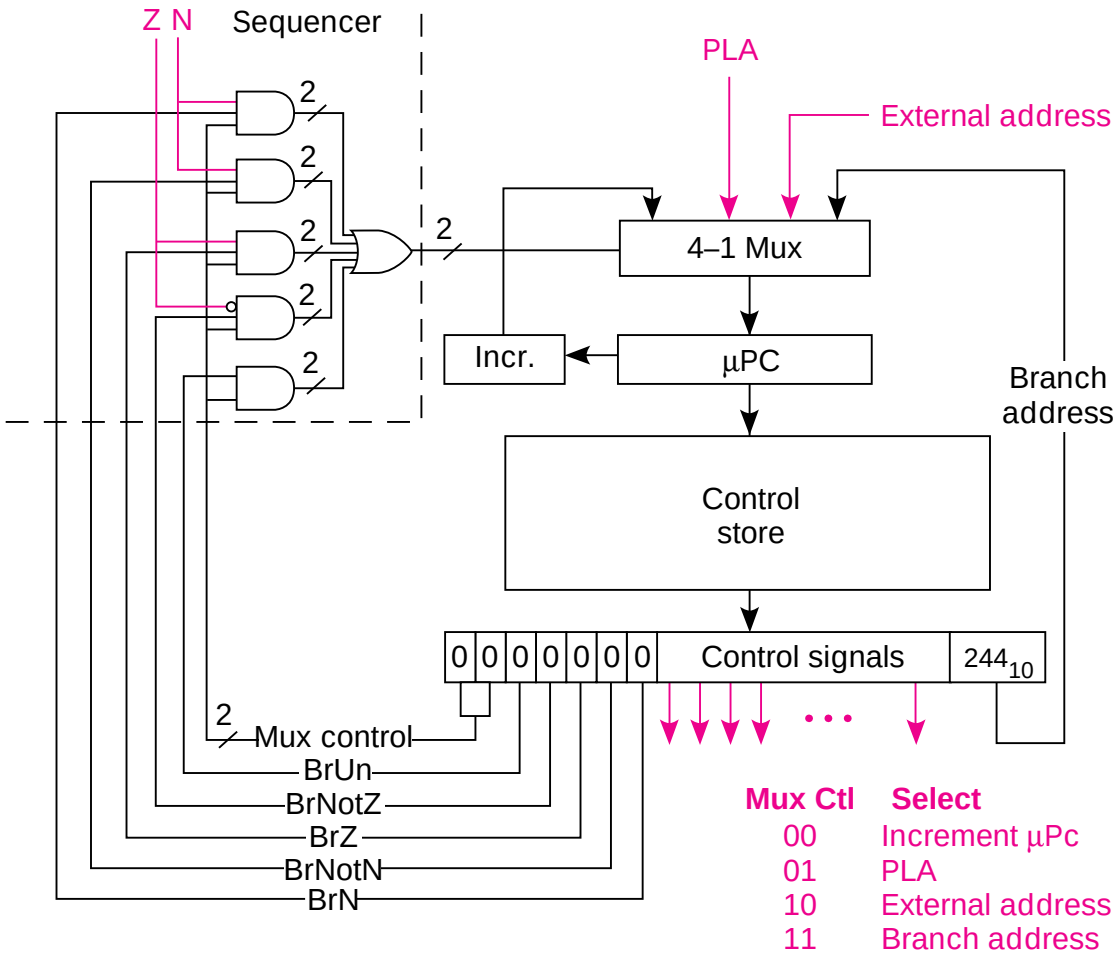
Tbl 5.2 Control Signals for the add Instruction

Address	Branch Control	PC_{out}	C_{out}	MD_{out}	R_{out}	MA_{in}	C_{in}	PC_{in}	IR_{in}	A_{in}	R_{in}	Inc4	Read	Wait	ADD	Gra	Grb	Grc	End
101	•••	1	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	0
102	•••	0	1	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
103	•••	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
200	•••	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0
201	•••	0	0	0	1	0	1	0	0	0	0	0	0	0	1	0	0	1	0
202	•••	0	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1

- **Addresses 101–103 are the instruction fetch**
- **Addresses 200–202 do the add**
- **Change of μcontrol from 103 to 200 uses a kind of μbranch**

© 1997 V. Heuring and H. Jordan

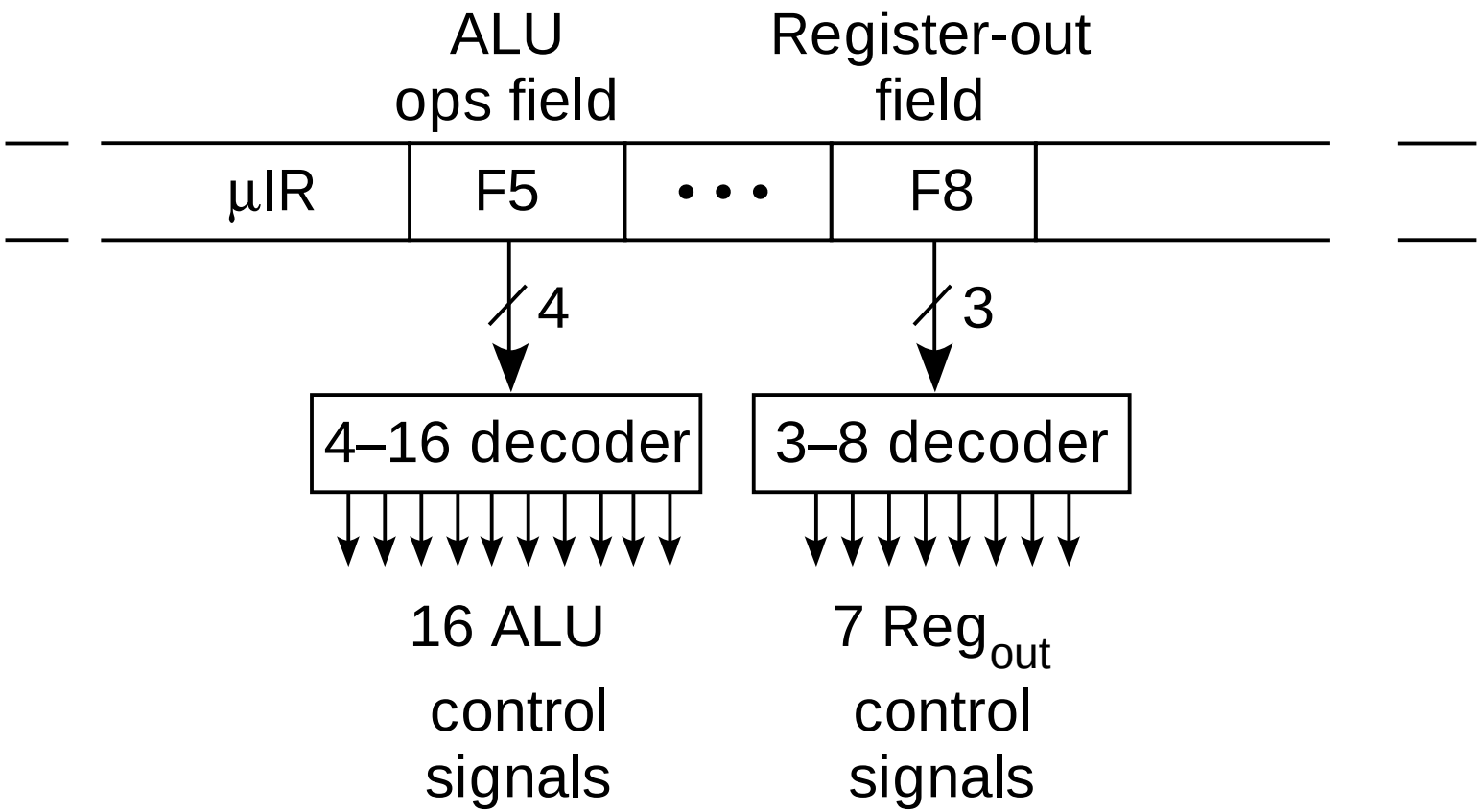
Fig 5.18 Branching Controls in the Microcoded Control Unit



- **5 branch conditions**
  - **NotN**
  - **N**
  - **NotZ**
  - **Z**
  - **Unconditional**
- **To 1 of 4 places**
  - **Next µinstruction**
  - **PLA**
  - **External address**
  - **Branch address**

© 1997 V. Heuring and H. Jordan

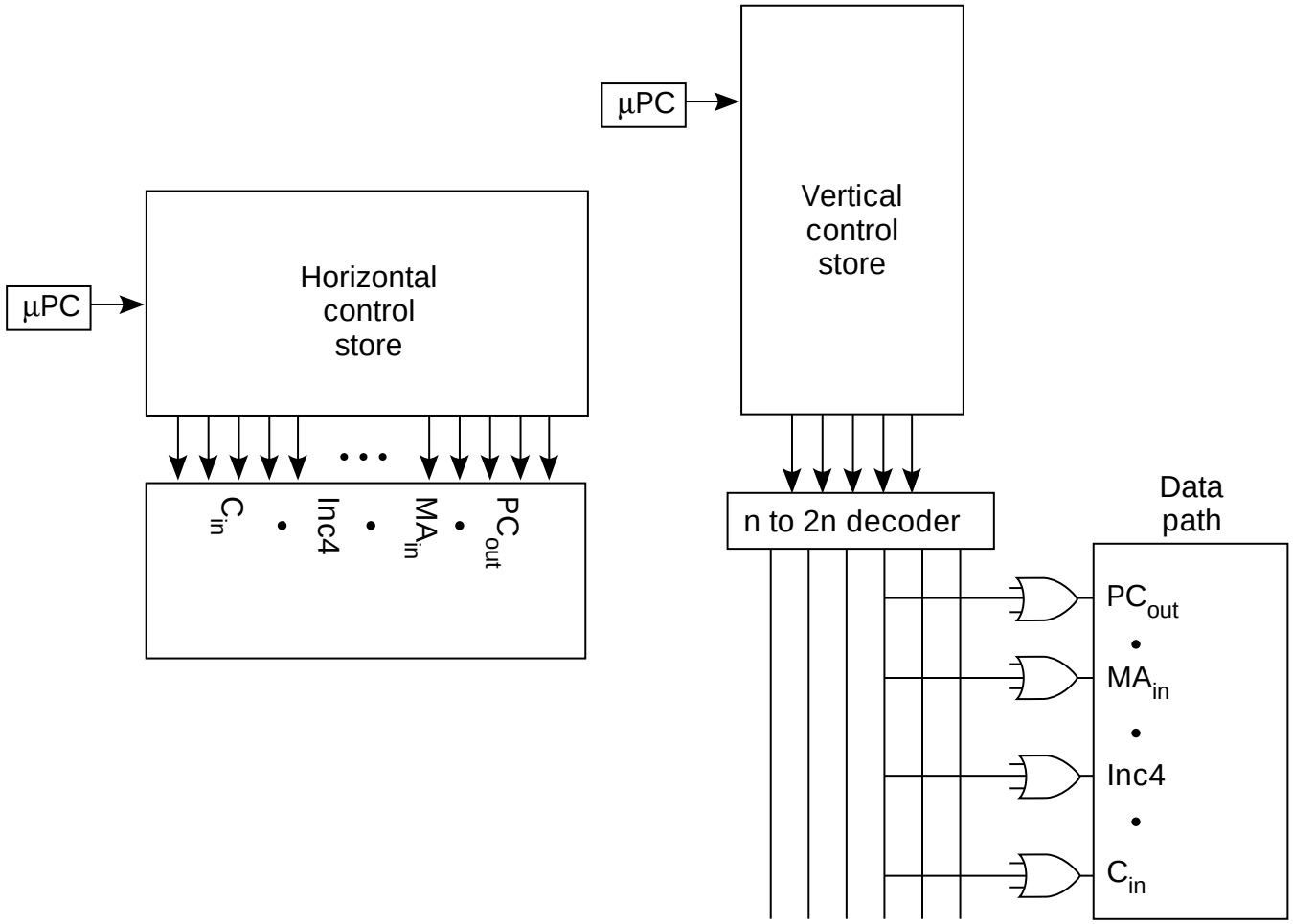
Fig 5.19 A Somewhat Vertical Encoding



- **Scheme would save (16 + 7) - (4 + 3) = 16 bits/word in the case illustrated**

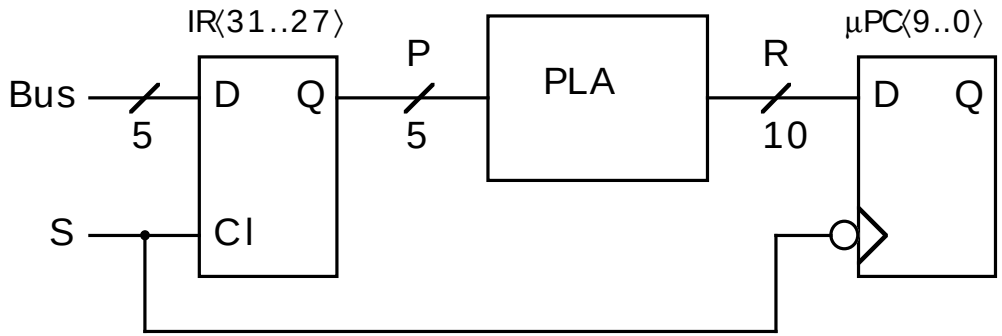
© 1997 V. Heuring and H. Jordan

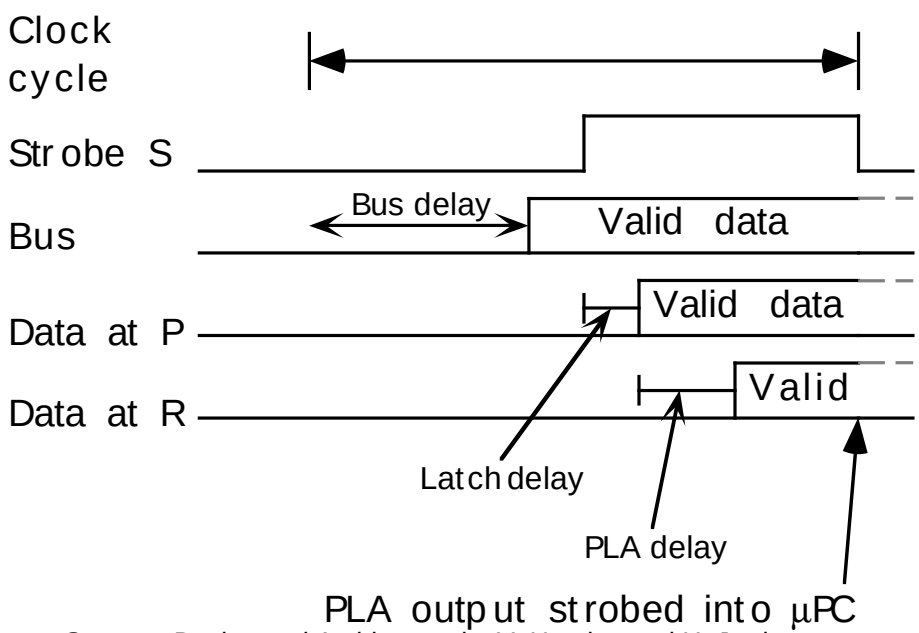
Fig 5.20 Completely Horizontal and Vertical Microcoding



© 1997 V. Heuring and H. Jordan

See-Through Latch Hardware for IR So μPC Can Load Immediately

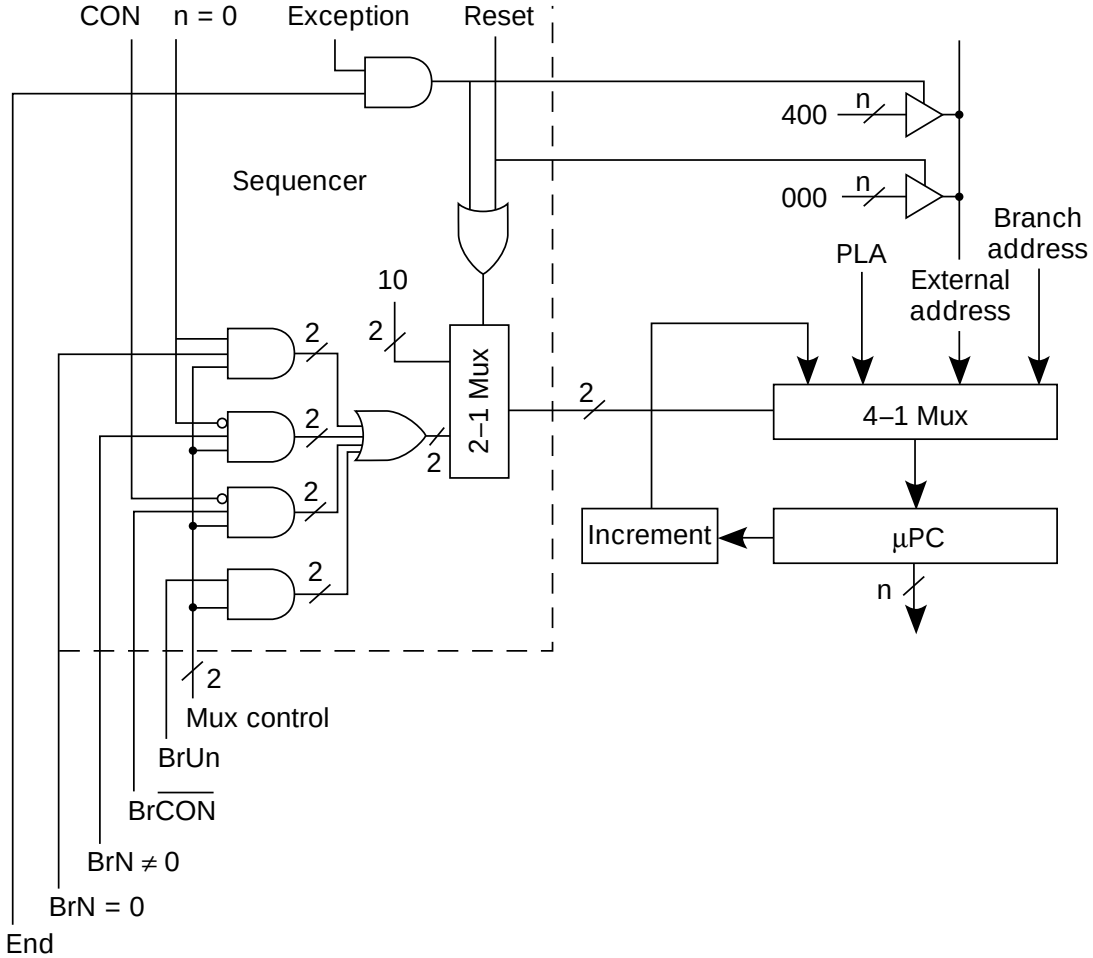




- **Data must have time to get from MD across Bus, through IR, through the PLA, and satisfy μPC set up time before trailing edge of S**

© 1997 V. Heuring and H. Jordan

Fig 5.21 SRC Microcode Sequencer



© 1997 V. Heuring and H. Jordan

Tbl 5.6 Somewhat Vertical Encoding of the SRC Microinstruction

F1	F2	F3	F4	F5	F6	F7	F8	F9
Mux Ctl	Branch control	End	Out signals	In signals	Misc.	Gate regs.	ALU	Branch address
00 01 10 11	000 BrUn 001 Br¬CON 010 BrCON 011 Br n=0 100 Br n≠0 101 None	0 Cont. 1 End	000 PC_{out} 001 C_{out} 010 MD_{out} 011 R_{out} 100 BA_{out} 101 $c1_{out}$ 110 $c2_{out}$ 111 None	000 MA_{in} 001 PC_{in} 010 IR_{in} 011 A_{in} 100 R_{in} 101 MD_{in} 110 None	000 Read 001 Wait 010 Ld 011 Decr 100 CON_{in} 101 C_{in} 110 Stop 111 None	00 Gra 01 Grb 10 Grc 11 None	0000 ADD 0001 C=B 0010 SHR 0011 Inc4 • • • 1111 NOT	10 bits
2 bits	3 bits	1 bit	3 bits	3 bits	3 bits	2 bits	4 bits	10 bits

© 1997 V. Heuring and H. Jordan