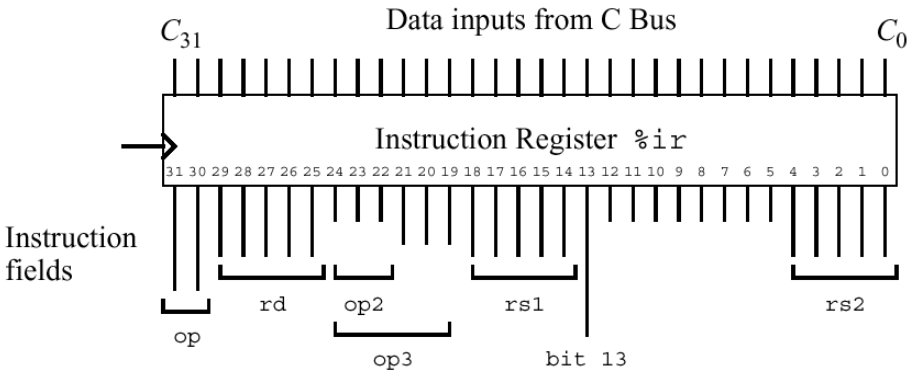
Chapter 5 Datapath and Control

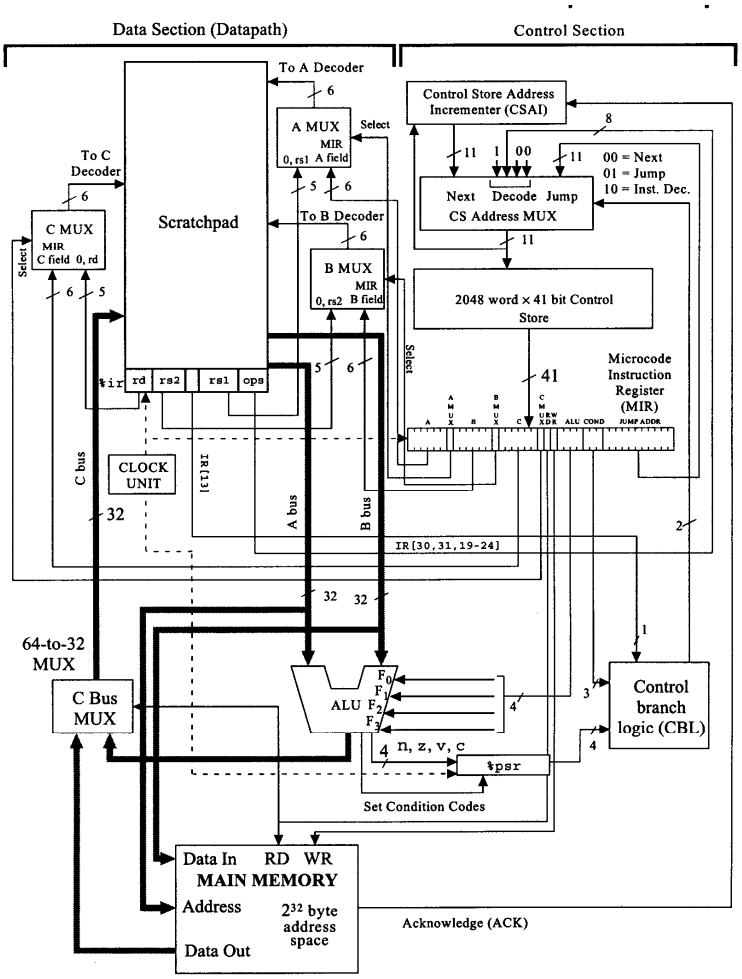
Example of ARC Instruction Format



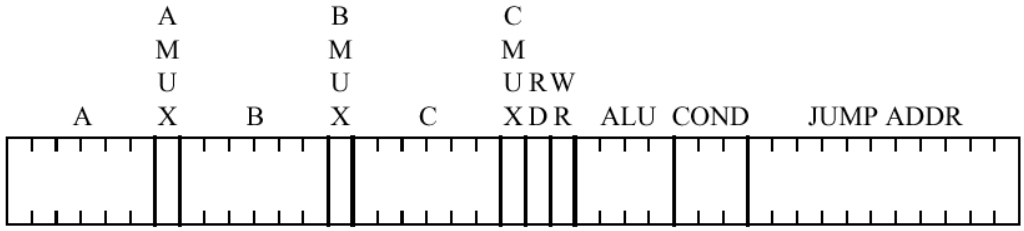
andcc %r1, %r2, %r3 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

1 0 0 0 0 1 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0

ARC Microprogram



A microword contains 41 bits comprising 11 fields as shown below.



- The A field determines which of the registers is to be placed on the A bus.
- The AMUX field selects whether the A decoder takes its input from the A field of the MIR (AMUX = 0) or from the rs1 field of %ir (AMUX = 1).
- The B, BMUX, and C, CMUX fields perform similar functions for the B and C buses. Since %r0 cannot be changed, the bit pattern 000000 can be used in the C field when none of these registers are to be written from the C bus.
- The RD and WR lines determine whether memory will be read or written respectively. A read takes place if RD = 1 and a write takes place if WR = 1. Both the RD and WR fields cannot be set to 1 at the same time, but both can be 0 if an instruction does not involve memory access.
- For both RD and WR, the memory address is taken from The A bus. For WR, the data input to memory is taken from the B bus, and for RD, the data output from memory is placed on the C bus.
- The RD line controls the 64-to-32 C bus MUX which determines whether the C bus is loaded from memory (RD = 1) or from the ALU (RD = 0).
- The ALU field determines which of the ALU operations is performed according to the settings in the figure shown below

F_3	F_2	F_1	F_0	Operation	Changes Condition Codes
0	0	0	0	ANDCC (A, B)	yes
0	0	0	1	ORCC (A, B)	yes
0	0	1	0	NORCC (A, B)	yes
0	0	1	1	ADDCC (A, B)	yes
0	1	0	0	SRL (A, B)	no
0	1	0	1	AND (A, B)	no
0	1	1	0	OR (A, B)	no
0	1	1	1	NOR (A, B)	no
1	0	0	0	ADD (A, B)	no
1	0	0	1	LSHIFT2 (A)	no
1	0	1	0	LSHIFT10 (A)	no
1	0	1	1	SIMM13 (A)	no
1	1	0	0	SEXT13 (A)	no
1	1	0	1	INC (A)	no
1	1	1	0	INCPC (A)	no
1	1	1	1	RSHIFT5 (A)	no

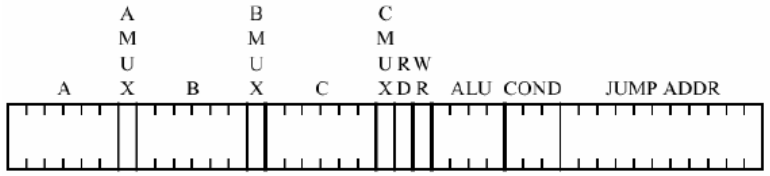
- The COND (conditional jump) field instructs the microcontroller to take the next microword from either the next control store location, or from the location in the JUMP ADDR field of the MIR, or from the opcode bits of the instruction in %ir. The COND field is interpreted according to the table below.

C_2 C_1 C_0	Operation
0 0 0	Use NEXT ADDR
0 0 1	Use JUMP ADDR if n = 1
0 1 0	Use JUMP ADDR if z = 1
0 1 1	Use JUMP ADDR if v = 1
1 0 0	Use JUMP ADDR if c = 1
1 0 1	Use JUMP ADDR if IR[13] = 1
1 1 0	Use JUMP ADDR
1 1 1	DECODE

- Finally, the JUMP ADDR field appears in the rightmost 11 bits of the microword. There are 2^{11} microwords in the control store, and so 11 bits are used in the JUMP ADDR field in order to jump to any instruction in the control store.

Microassembly Language Translation

```
0: R[ir] ← AND(R[pc],R[pc]); READ;
1: DECODE;
```



0 10000001000000100101010010101000

%r32 %r32 %r37 AND Next

1 10000001000000100000000001011111

The bits in the fields of the 41-bit microword can be filled in as shown in the above first statement.

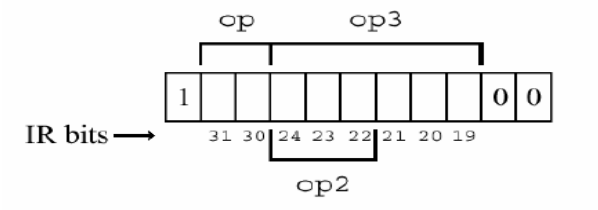
- The PC (R[32]) is placed on both the A and B busses for an AND operation (code 0101)
- The RD bit is set to 1 and the WR bit is 0.
- The AMUX and BMUX fields are both 0, since the input to the A and B decoders comes from the MIR A and B fields rather than from registers specified in a machine language instruction.
- The destination register for the READ is the IR (register 37). The CMUX field is also 0, since the identity of the destination register is in this microcode instruction rather than in the rd field of a machine language instruction.
- The COND field is 000 since the next microinstruction will be executed rather than jumping to a non-adjacent line of the microprogram. The JUMP ADDR field does not matter in this case and is arbitrarily filled with 0's.

The second microword implements the 256-way branch. For this case, all that matters is that the bit pattern 111 appears in the COND field for the DECODE operation and that no registers, memory locations, or condition codes

are disturbed. In the text all bits for the word are 0's with the exception of 111 for COND and 0101 for the ALU OPCODE. This corresponds to the AND operation. Any other ALU operation that does not affect the condition codes can also be used.

The bit pattern 111 is used in the COND field when an instruction is being decoded. When the COND field is 111, then the next instruction is taken from a location that is generated by appending 1 to the left of bits 31 and 30 (op) of the instruction, then appending 00 to the right of bits 24 – 19 (op3).

Decoding example



addcc %r1, %r2, %r3

op	rd	op3	rs1	13	rs2
31 30	29 28 27 26 25	24 23 22 21 20 19	18 17 16 15 14	13	12 11 10 9 8 7 6 5 4 3 2 1 0
1 0	0 0 0 1 1	0 1 0 0 0 0	0 0 0 0 1	0	0 0 0 0 0 0 0 0 0 0 0 1 0

$$(1\ 10\ 010000\ 00)_2 = (1600)_{10}$$