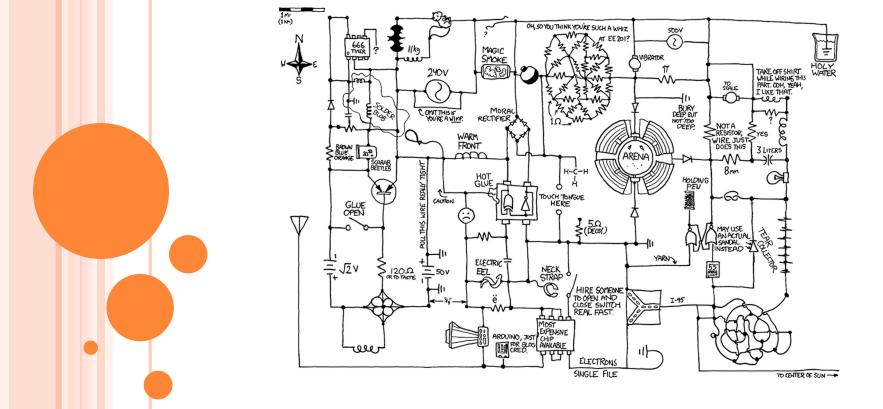
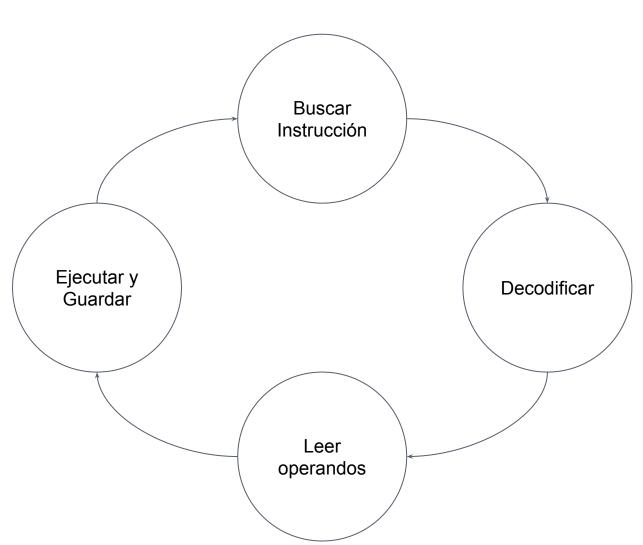
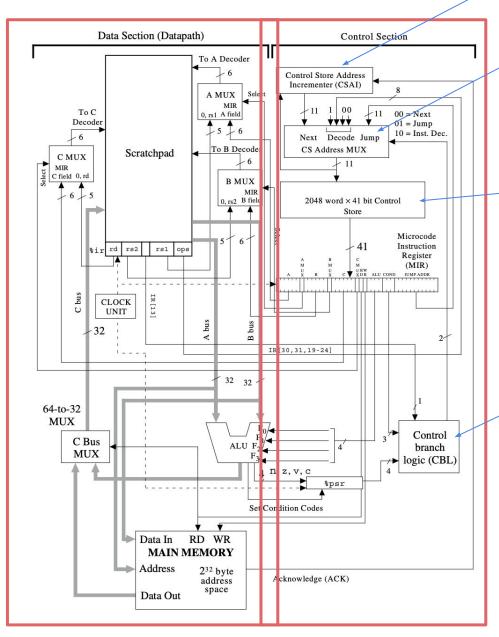
MICROARQUITECURA



CICLO DE BÚSQUEDA



Introducción



Calcula la siguiente micro instrucción

> Multiplexor para acceder a la memoria que almacena el micro código

> > Memoria que almacena el micro código

Lógica de control de saltos

EJERCICIO

En un procesador ARC el registro Program Counter apunta a la siguiente instrucción guardada en RAM:

addcc %r10, 48, %r1

Detallar los pasos de microprograma que la decodifican y para cada uno de ellos indicar los valores presentes en las entradas y en las salidas de cada uno de los siguientes bloques funcionales:

- multiplexor de direcciones de la memoria de control
- incrementador de direcciones de la memoria de control
- decodificadores de los buses A B y C
- multiplexores que intervienen en la decodificación de los buses A B y C
- multiplexor de datos del bus C
- bus de datos A B y C
- entrada de direcciones del módulo de memoria RAM

Considere que antes de la ejecución de esta instrucción %r10 = %r1 = 0

Buscar Instrucción

	op 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
SETHI Format	0 0 rd op2 imm22
Branch Format	0 0 0 cond op2 disp22
CALL format	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 0 1 disp30
	i
	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
Arithmetic	1 0 rd op3 rs1 0 0 0 0 0 0 0 0 0 rs2
Formats	1 0 rd op3 rs1 1 simm13
	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
	1 1 rd op3 rs1 0 0 0 0 0 0 0 0 rs2
Memory Formats	1 1 rd op3 rs1 1 simm13

op	Format
00	SETHI/Branch
01	CALL
10	Arithmetic
11	Memory

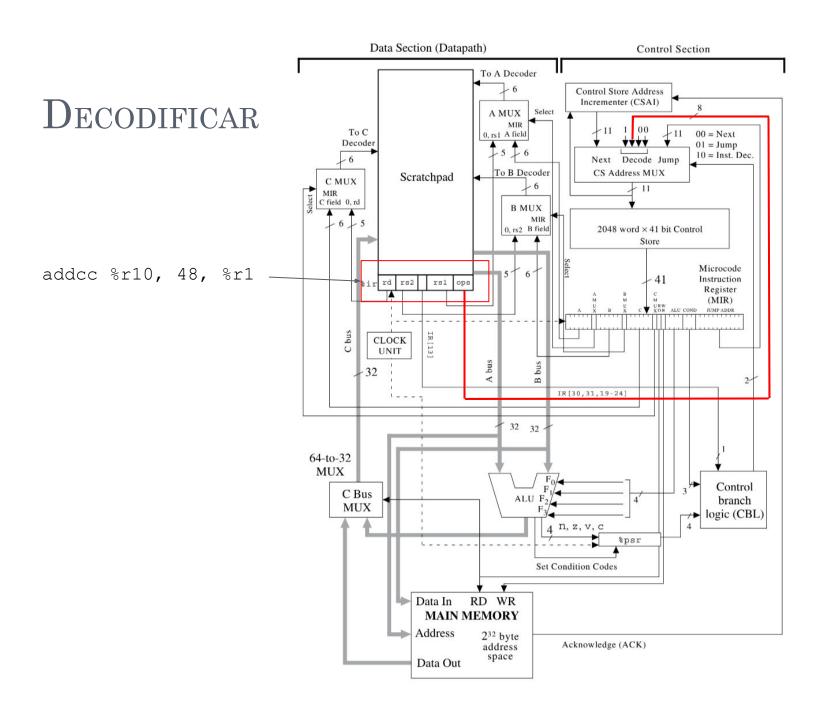
op2	Inst.
010	branch
100	sethi

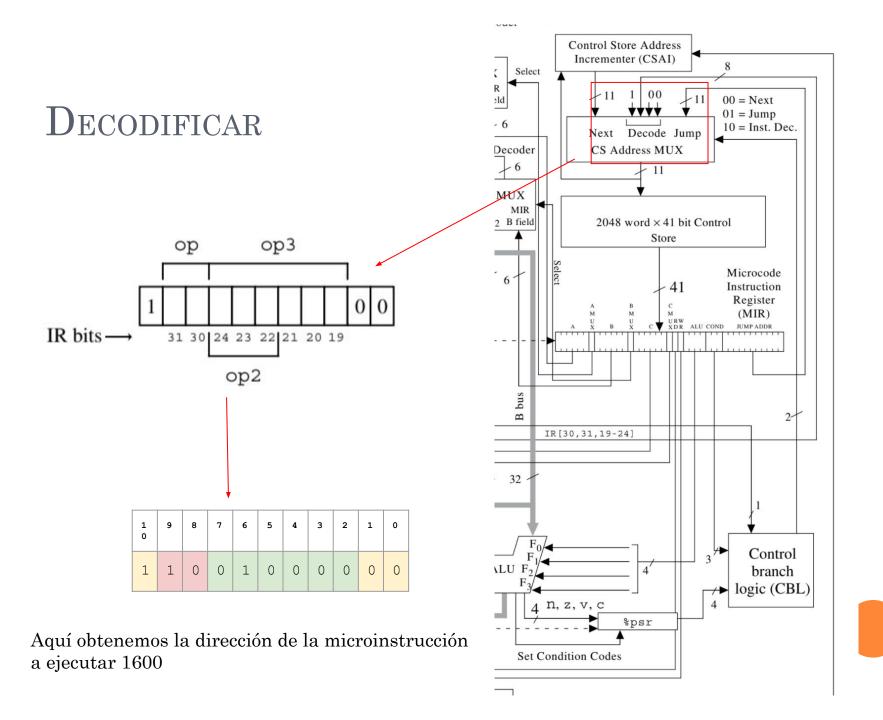
op3 (o	p=10)
010000	addcc
010001	andcc
010010	orcc
010110	orncc
100110	srl
111000	jmpl

p3 (op=11)	cond	branch
00000 ld	0001	be
00100 st	0101	bcs
	0110	bneg
	0111	bvs
	1000	ba

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	0
R	13		,						n	z	v	С		8 9				Ui - 3														
													- 1							-										7		_

3	3	2 9	2		2	2 5		2		2	2	1 9	1 8	1 7			1 4	1 3	1 2	1	1	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0





Ejecución

```
/ addcc 

1600: IF R[IR[13]] THEN GOTO 1602; / Is second source operand immediate? 

1601: R[rd] \leftarrow ADDCC(R[rs1],R[rs2]); / Perform ADDCC on register sources 

GOTO 2047; / Get sign extended simm13 field 

1603: R[rd] \leftarrow ADDCC(R[rs1],R[temp0]); / Perform ADDCC on register/simm13 

GOTO 2047; / sources
```

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

F_3	F_2	F_1	F_0	Operation
0	0	0	0	ANDCC (A, B)
0	0	0	1	ORCC (A, B)
0	0	1	0	NORCC (A, B)
0	0	1	1	ADDCC (A, B)
0	1	0	0	SRL (A, B)
0	1	0	1	AND (A, B)
0	1	1	0	OR (A, B)
0	1	1	1	NOR (A, B)
1	0	0	0	ADD (A, B)
1	0	0	1	LSHIFT2 (A)
1	0	1	0	LSHIFT10 (A)
1	0	1	1	SIMM13 (A)
1	1	0	0	SEXT13 (A)
1	1	0	1	INC (A)
1	1	1	0	INCPC (A)
1	1	1	1	RSHIFT5 (A)

		A		В		C				1
		M		M		M				1
		U		U		URW				
	A	X	В	X	C	XDR	ALU C	COND	JUMP ADDR	
		П		П						T
Ц		Ш		Ш		Ш				

Ejecución

	U	U	1	URW		
A	X B	X	C Z	XDR	ALU COND	JUMP ADDR
00000	00000	000000	0000	0000	000010111	1001000010
00000	010000	0001000	000	1000	001111011	111111111
10010	10000	0000100	001	0001	1000000	000000000
00000	011000	0010000	0000	1000	001111011	111111111

C

В

A

	F_3	F_2	r_1	F_0	Operation
	0	0	0	0	ANDCC (A, B)
-	0	0	0	1	ORCC (A, B)
-	0	0	1	0	NORCC (A, B)
-	0	0	1	1	ADDCC (A, B)
-	0	1	0	0	SRL (A, B)
-	0	1	0	1	AND (A, B)
-	0	1	1	0	OR (A, B)
-	0	1	1	1	NOR (A, B)
-	1	0	0	0	ADD (A, B)
-	1	0	0	1	LSHIFT2 (A)
-	1	0	1	0	LSHIFT10 (A)
-	1	0	1	1	SIMM13 (A)
-	1	1	0	0	SEXT13 (A)
	1	1	0	1	INC (A)
	1	1	1	0	INCPC (A)
	1	1	1	1	RSHIFT5 (A)

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

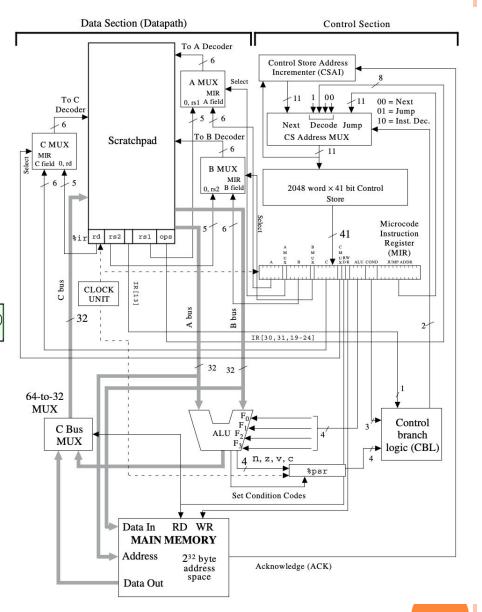
DECODIFICAR

00	A 0 0 0	A M U X	E		B M U X		C 0 0	00	X	R	R		AL	0 0	_	01	1	J 10		1P 4	П	DR 	10
00	000	01	000	000	1	000	00	00	1	0	0	0	01	1	1:	10	1	11	1:	11	1	11	11
10	010	10	000	000	0	10	00	01	0	0	0	1	10	0 0	0	00	0	00	0	0 0	0	0 0	000
00	000	01	100	001	0	000	00	00	1	0	0	0	01	1	1:	10	1	11	1	11	1	11	.11

addcc %r10, 48, %r1

1600: IF R[IR[13]] THEN GOTO 1602;

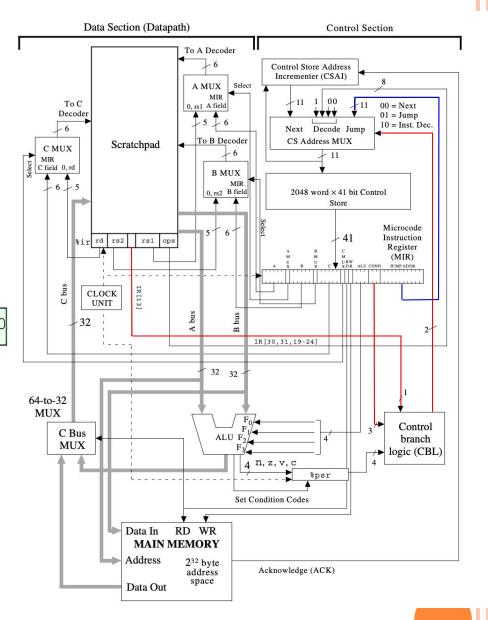
		Α		В		C					
		M	[M		N	1				
		U		U		U	R	W			
	A	X	В	X	C	X	D	R	ALU	COND	JUMP ADDR
	111			Ţ		Ţ	П	П			
 0 0	000	0 0	0000	0 0 0	 00000	000	101	IJΙ	0000	101	11001000010
							П		1 1 1		



addcc %r10, 48, %r1

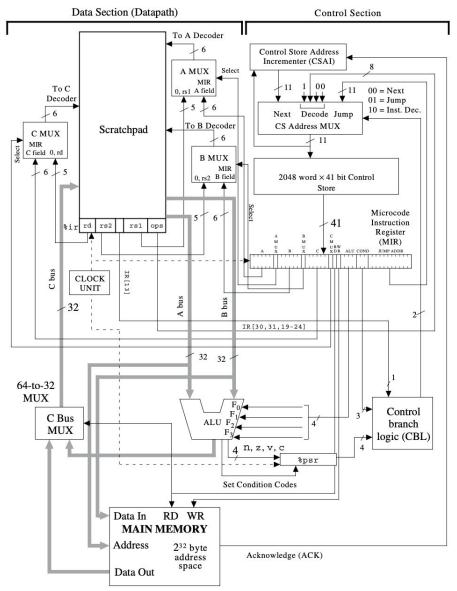
1600: IF R[IR[13]] THEN GOTO 1602;

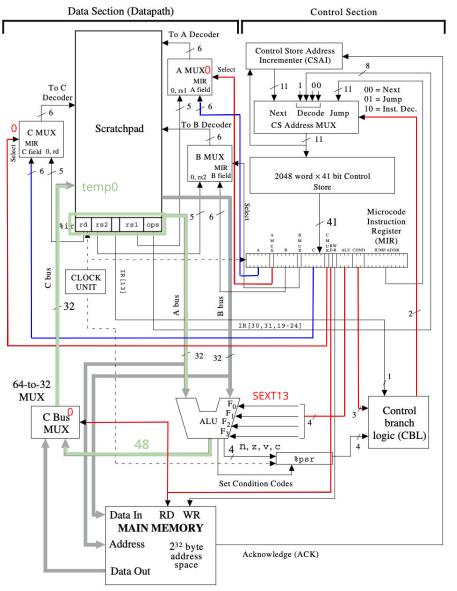
		Α		В		C				
		M	[M		M	[
		U		U		U	RW	7		
	A	X	В	X	C	X	DR	ALU	COND	JUMP ADDR
000	000	00	0000	000	0000	000	00	0000	101	11001000010
	1.1.1			1					1.1	



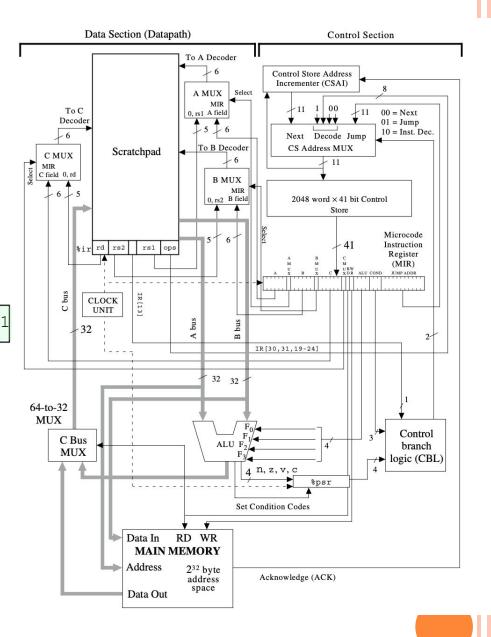
```
1602: R[temp0] ← SEXT13(R[ir]);

A B C
M M M M
U U URW
A X B X C XDR ALU COND JUMP ADDR
```

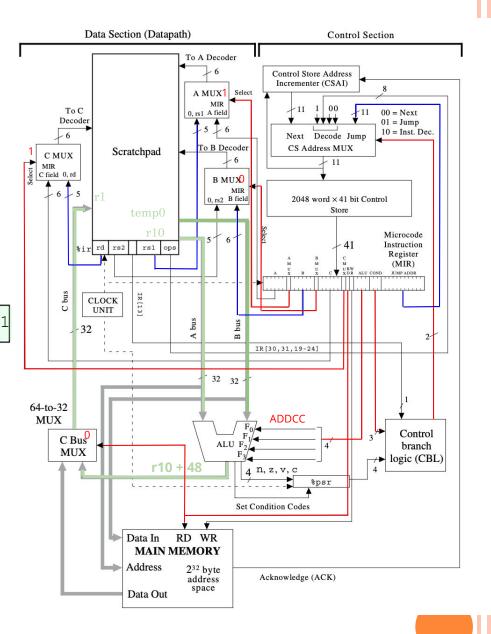




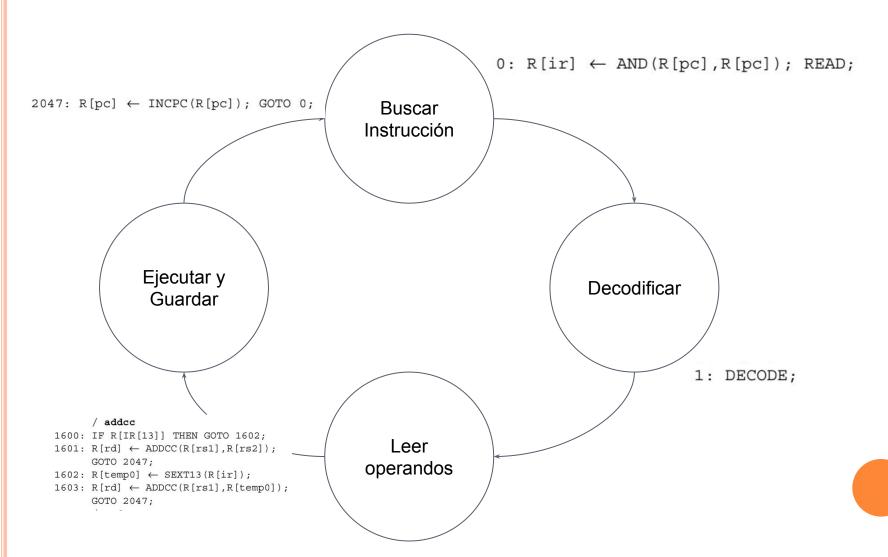
```
1603: R[rd] \leftarrow ADDCC(R[rs1], R[temp0]);
       GOTO 2047;
               В
                         C
    A
    M
              M
                         M
    U
               U
                         URW
    X
               X
A
                         XDR ALU COND
                                          JUMP ADDR
```



```
1603: R[rd] \leftarrow ADDCC(R[rs1], R[temp0]);
       GOTO 2047;
               В
                         C
    A
    M
              M
                         M
    U
               U
                         URW
    X
A
               X
                         XDR ALU COND
                                          JUMP ADDR
```



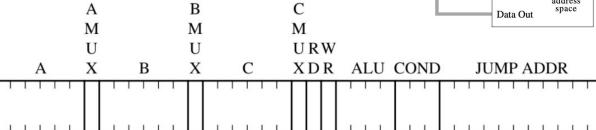
Ciclo de búsqueda

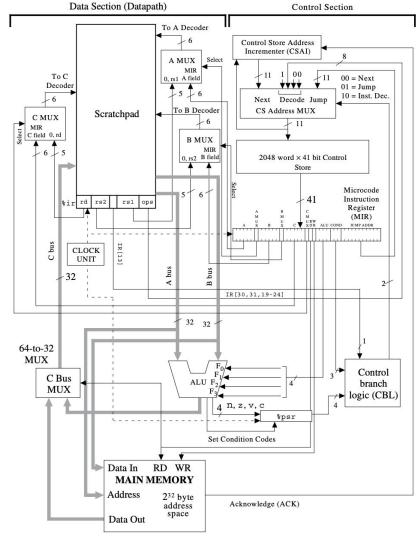


Buscar Instrucción

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

2047: $R[pc] \leftarrow INCPC(R[pc]); GOTO 0;$





Preguntas