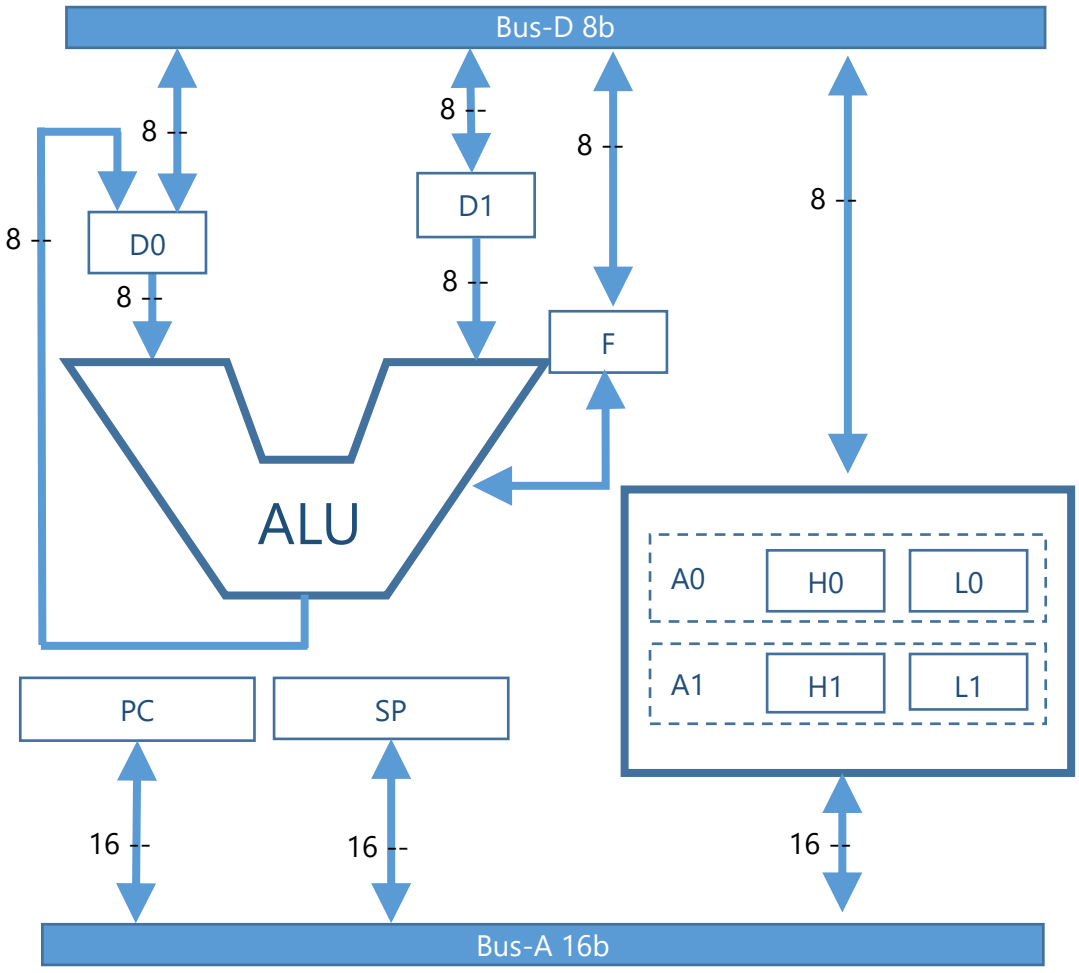
CH-2020

Arquitectura

El CH-2020 es una CPU virtual de arquitectura von Newmann. Su bus de direcciones es de 16 bits mientras que el de datos lo es de 8 bits.



La unidad de control tiene dos registros de datos (D0 y D1), dos de direcciones (A0 y A1), un contador de programa (PC), un puntero de pila (SP) y un registro de flags (F) que registra los resultados de las operaciones en la ALU.

D0 hace las funciones de primer operando de la ALU y acumulador.

D1 hace las funciones de segundo operando.

Los registros Ax tienen 16 bits y se puede acceder a cada uno de sus bytes independientemente, en el caso de A0: H0 y L0 y en el caso de A1: H1 y L1, siendo Hx el MSB y el Lx el LSB.

La ALU permite operaciones simples con D0 y en su caso D1

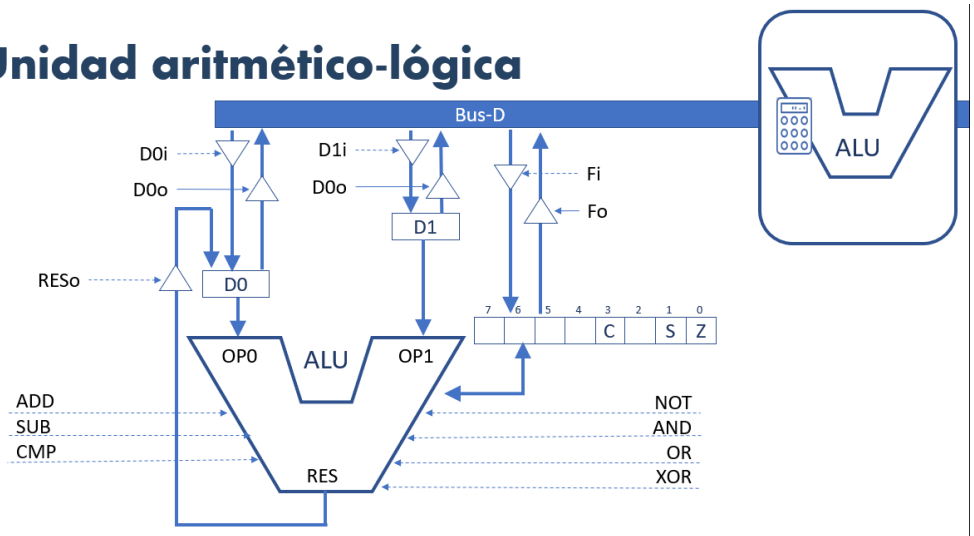
- Aritméticas: NEG, ADD, SUB, CMP.
- Lógicas: NOT, AND, OR, XOR.
- Deslizamiento: SL, SR.

Las funciones de transferencia de datos (MOV) utilizan como modos de direccionamiento: Inmediato, Registro, Directo, Indirecto por registro.

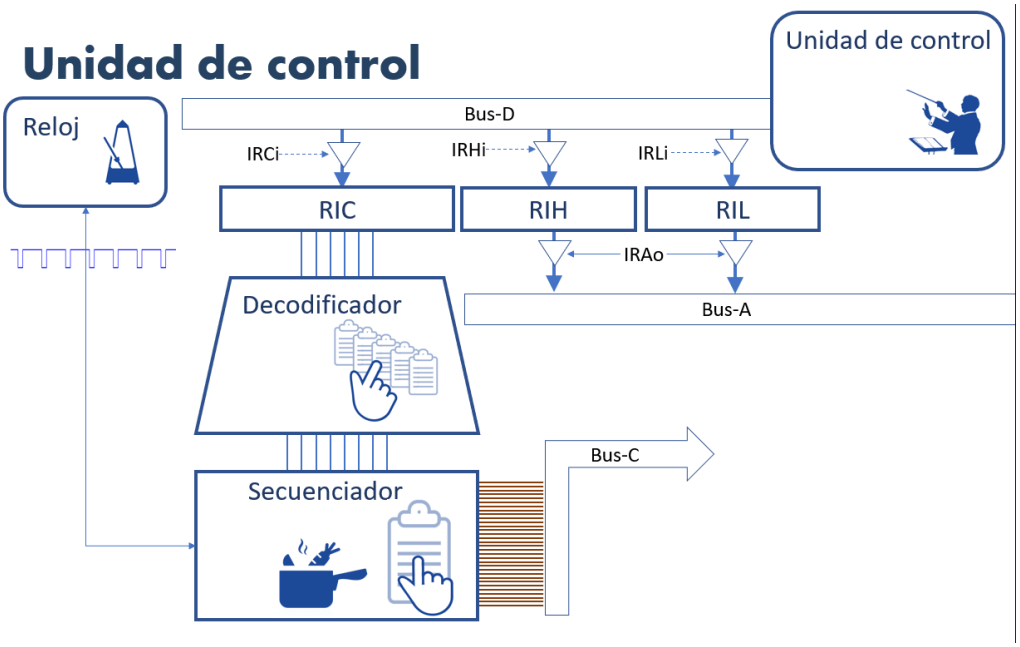
Las opciones de control de flujo de salto incondicional absoluto (JMP) y saltos relativos condicionales (JR), llamadas a subrutinas CALL y RET.

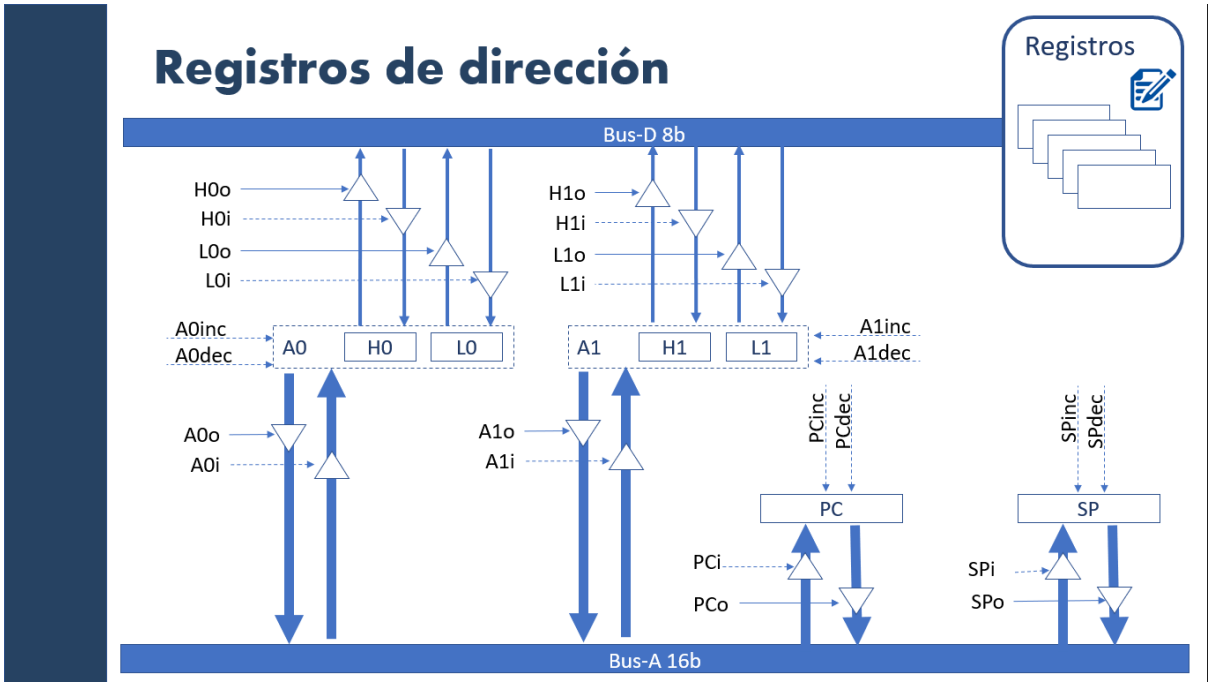
Organización

Unidad aritmético-lógica



Unidad de control





Repertorio

Tabla 1: Codificación de los registros de datos

r	rrr
D0	0b000
D1	0b001
F	0b011
H0	0b100
L0	0b101
H1	0b110
L1	0b111

Tabla 2: Codificación de los registros de direcciones

a	aa
A0	0b00
A1	0b01
SP	0b11
PC	0b11

Tabla 3: Codificación de las condiciones de salto

c	ccc
C	0b000
NC	0b001
Z	0b010
NZ	0b011
S	0b100
NS	0b101
O	0b110
NO	0b111

Repertorio de instrucciones

NOP **N**o **OP**eration

Descripción: No realizar ninguna acción

Formato: NOP

Código: 0x00

Operación

Flags No afectados

0	0	0	0	0	0	0	0

STOP **STOP**

Descripción: Detener el procesador

Formato: STOP

Código: 0x01

Operación Detener el procesador

Flags No afectados

0	0	0	0	0	0	0	1

CLC **CL**ear **C**arry flag

Descripción: Poner a cerro el flag de acarreo

Formato: CLC

Código: 0x02

Operación C ← 0

Flags C

0	0	0	0	0	0	1	0

STC **S**e**T** **C**arry flag

Descripción: Poner a uno el flag de acarreo

Formato: STC

Código: 0x03

Operación C ← 1

Flags C

0	0	0	0	0	0	1	1

ADD **ADD**

Descripción: Sumar

Formato: ADD

Código: 0x08

0	0	0	0	1	0	0	0

Operación D0 ← (D0) + (D1)

Flags C O S Z

SUB **SUB**stract

Descripción: Restar

Formato: SUB

Código: 0x09

0	0	0	0	1	0	0	1

Operación D0 ← (D0) – (D1)

Flags C O S Z

CMP **C**o**MP**are

Descripción: Comparar

Formato: CMP

Código: 0x0A

0	0	0	0	1	0	1	0

Operación (D0) – (D1)

Flags C O S Z

NEG **NEG**ative

Descripción: Negativo / Complemento a 2

Formato: NEG

Código: 0x0B

0	0	0	0	1	0	1	1

Operación D0 ← -(D0)

Flags C=0, O S Z

NOT **NOT**

Descripción: Invertir / Complemento a 1

Formato: NOT

Código: 0x0C

0	0	0	0	1	1	0	0

Operación D0 ← /(D0)

Flags C=0 O S Z

AND **AND**

Descripción: Y lógico bit a bit

Formato: AND

Código: 0x0D

0	0	0	0	1	0	1	1

Operación D0 ← (D0) ∧ (D1)

Flags C O S Z

OR **OR**

Descripción: O lógico bit a bit

Formato: OR

Código: 0x0E

0	0	0	0	1	1	1	0

Operación D0 ← (D0) ∨ (D1)

Flags C=0 O S Z

XOR e**X**clusive **OR**

Descripción: O exclusivo bit a bit

Formato: XOR

Código: 0x0F

0	0	0	0	1	1	1	1

Operación D0 ← (D0) ⊕ (D1)

Flags C O S Z

SL **S**hift **L**eft

Descripción: Deslizar a la izquierda 1 bit

Formato:

Código: 0x10

0	0	0	1	0	0	0	0

Operación C = $D0_7$, D0 ← (D0) << 1

Flags C O S Z

SR **S**hift **R**ight

Descripción: Deslizar a la derecha 1 bit

Formato: SR

Código: 0x11

0	0	0	1	0	0	0	1

Operación C= $D0_0$, D0 ← (D0) >> 1

Flags C O S Z

JMP **J**u**MP**

Descripción: Romper el flujo de la ejecución.

Formato: JMP add

Código: 0x20, addh, addl

add = addh*0x100+addl

PC ← add

Flags No afectados

0	0	1	0	0	0	0	0

JR **J**ump **R**elative

Descripción: Romper el flujo de la ejecución.

Formato: JR A dsp Incondicional

Código: 0x21 dsp

dsp: byte con signo

PC ← (PC) + dsp

Flags No afectados

0	0	1	0	0	0	0	1

Formato: JR ccc dsp Incondicional

Código: cop dsp cop = 0x28 -0x2F

dsp: byte con signo

Operación PC ← (PC) + dsp

ccc Ver Tabla 3

Flags No afectados

0	0	1	0	1	c	c	c

CALL **CALL**

Descripción: Llamada a subrutina.

Formato: CALL add

Código: 0x22 addh addl

add = addh*0x100+addl

Operación SP--, (SP) ← (PC_{lsb})
SP--, (SP) ← (PC_{msb})
PC ← add

Flags No afectados

0	0	1	0	0	0	1	0

RET **RET**urn

Descripción: Retorno de subrutina

Formato: RET

Código: 0x23

0	0	0	1	0	0	1	1

Operación $PC_{msb} \leftarrow (SP)$, SP++
$PC_{lsb} \leftarrow (SP)$, SP++

Flags No afectados

PUSH **PUSH** byte on stack

Descripción: Apilar registro de datos

Formato: PUSH r

Código: cop cop = 0x30 - 0x37

0	0	1	1	0	r	r	r

r: rrr Ver Tabla 1

Operación $SP\text{--}, (SP) \leftarrow (x_{lsb})$
$SP\text{--}, (SP) \leftarrow (x_{msb})$

Flags No afectados

POP **POP** byte from stack

Descripción: Extraer de la pila registro de datos

Formato: POP r

Código: cop cop = 0x38 – 0x3F

0	0	1	1	1	r	r	r

r: rrr Ver Tabla 1

Operación $x_{msb} \leftarrow (SP)$, SP++
$x_{lsb} \leftarrow (SP)$, SP++

Flags No afectados

DEC **DEC**rement

Descripción: Decrementar un registro de direcciones

Formato: DEC a

Código: cop cop = 0x48 – 0x4B

0	1	0	0	1	0	a	a

a: aa Ver Tabla 2

Operación $a \leftarrow (a)-1$

Flags No afectados

INC **INC**rement

Descripción:	Incrementar registro de direcciones		
Formato:	INC a		
Código:	cop	cop = 0x4C – 0x4F	0 1 0 0 1 1 a a
a: aa	Ver Tabla 2		
Operación	a ← (a)+1		
Flags	No afectados		

MOV **MOV**e

Descripción:	Mover datos		
Formato:	MOV #val r		
Código:	cop val	cop = 0x50 – 0x57	0 1 0 1 0 r r r
r: rrr	Ver Tabla 1		
Operación	r ← val		
Flags	No afectados		
Formato:	MOV r dir		
Código:	cop dirh dirl	cop = 0x60 – 0x67	0 1 1 0 0 r r r
dir	dirh*0x100 + dirl		
r: rrr	Ver Tabla 1		
Operación	r ← (dir)		
Flags	No afectados		
Formato:	MOV dir r		
Código:	cop dirh dirl	cop = 0x67 – 0x6F	0 1 1 0 1 r r r
dir	dirh*0x100 + dirl		
r: rrr	Ver Tabla 1		
Operación	(dir) ← r		
Flags	No afectados		
Formato:	MOV a y	Mover 16 bits	
Código:	cop	cop = 0x70 – 0x7F	0 1 1 1 a a y y
Operación	y ← a		
a: aa	Ver Tabla 2		
y: yy	Igual que a: aa		
Flags	No afectados		

MOV **MOV**e

Formato: MOV r s

Código: cop — cop = 0x80 – 0xBF

Operación: s ← r

r: rrr — Ver Tabla 1

s: sss — Igual que r: rrr

Flags: No afectados

Formato: MOV r (a)

Código: cop — cop = 0xC0 – 0xDF

Operación: (a) ← r

r: rrr — Ver Tabla 1

a: aa — Ver Tabla 2

Flags: No afectados

Formato: MOV (a) r

Código: cop — cop = 0xE0 – 0xFF

Operación: r ← (a)

r: rrr — Ver Tabla 1

a: aaa — Ver Tabla 2

Flags: No afectados

Tabla resumen

OJO en la siguiente tabla el nibble más significativo define la columna y el menos significativo define la fila, de este modo el código de operación se obtiene encadenando columna + fila.

LSN \ MSN	0	1	2	3	4	5	6	7
0	NOP	SL	JMP dir	PUSH D0		MOV #val D0	MOV dir D0	MOV A0 A0
1	STOP	SR	JR A dsp	PUSH D1		MOV #val D1	MOV dir D1	MOV A0 A1
2	CLC		CALL dir					MOV A0 SP
3	STC		RET	PUSH F		MOV #val F	MOV dir F	MOV A0 PC
4				PUSH L0		MOV #val L0	MOV dir L0	MOV A1 A0
5				PUSH H0		MOV #val H0	MOV dir H0	MOV A1 A1
6				PUSH L1		MOV #val L1	MOV dir L1	MOV A1 SP
7				PUSH H1		MOV #val H1	MOV dir H1	MOV A1 PC
8	ADD		JR C dsp	POP D0	DEC A0		MOV D0 dir	MOV SP A0
9	SUB		JR NC dsp	POP D1	DEC A1		MOV D1 dir	MOV SP A1
A	CMP		JR Z dsp		DEC SP			MOV SP SP
B	NEG		JR NZ dsp	POP F	DEC PC		MOV F dir	MOV SP PC
C	NOT		JR S dsp	POP L0	INC A0		MOV L0 dir	MOV PC A0
D	AND		JR NS dsp	POP H0	INC A1		MOV H0 dir	MOV PC A1
E	OR			POP L1	INC SP		MOV L1 dir	MOV PC SP
F	XOR			POP H1	INC PC		MOV H1 dir	MOV PC PC

LSN \ MSN	8	9	A	B	C	D	E	F
0	MOV D0 D0		MOV L0 D0	MOV L1 D0	MOV D0 (A0)	MOV L0 (A0)	MOV (A0) D0	MOV (A0) L0
1	MOV D0 D1		MOV L0 D1	MOV L1 D1	MOV D0 (A1)	MOV L0 (A1)	MOV (A1) D0	MOV (A1) L0
2					MOV D0 (SP)	MOV L0 (SP)	MOV (SP) D0	MOV (SP) L0
3	MOV D0 F		MOV L0 F	MOV L1 F	MOV D0 (PC)	MOV L0 (PC)	MOV (PC) D0	MOV (PC) L0
4	MOV D0 L0		MOV L0 L0	MOV L1 L0	MOV D1 (A0)	MOV H0 (A0)	MOV (A0) D1	MOV (A0) H0
5	MOV D0 H0		MOV L0 H0	MOV L1 H0	MOV D1 (A1)	MOV H0 (A1)	MOV (A1) D1	MOV (A1) H0
6	MOV D0 L1		MOV L0 L1	MOV L1 L1	MOV D1 (SP)	MOV H0 (SP)	MOV (SP) D1	MOV (SP) H0
7	MOV D0 H1		MOV L0 H1	MOV L1 H1	MOV D1 (PC)	MOV H0 (PC)	MOV (PC) D1	MOV (PC) H0
8	MOV D1 D0	MOV F D0	MOV H0 D0	MOV H1 D0		MOV L1 (A0)		MOV (A0) L1
9	MOV D1 D1	MOV F D1	MOV H0 D1	MOV H1 D1		MOV L1 (A1)		MOV (A1) L1
A						MOV L1 (SP)		MOV (SP) L1
B	MOV D1 F	MOV F F	MOV H0 F	MOV H1 F		MOV L1 (PC)		MOV (PC) L1
C	MOV D1 L0	MOV F L0	MOV H0 L0	MOV H1 L0	MOV F (A0)	MOV H1 (A0)	MOV (A0) F	MOV (A0) H1
D	MOV D1 H0	MOV F H0	MOV H0 H0	MOV H1 H0	MOV F (A1)	MOV H1 (A1)	MOV (A1) F	MOV (A1) H1
E	MOV D1 L1	MOV F L1	MOV H0 L1	MOV H1 L1	MOV F (SP)	MOV H1 (SP)	MOV (SP) F	MOV (SP) H1
F	MOV D1 H1	MOV F H1	MOV H0 H1	MOV H1 H1	MOV F (PC)	MOV H1 (PC)	MOV (PC) F	MOV (PC) H1