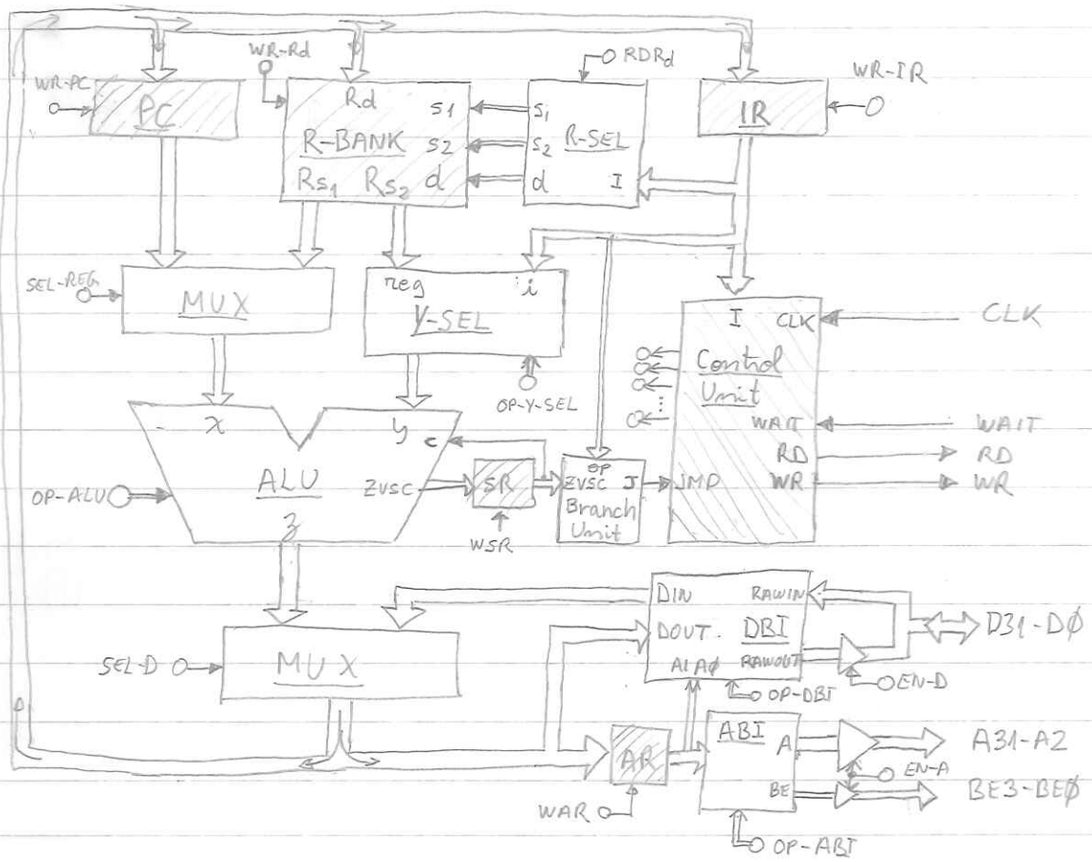
La CPU



En el circuito se pueden distinguir:

+ unidades de almacenamiento : ▨ PC, R-BANK, etc.

+ " " combinacionales : ▭ ALU, MUX, etc.

+ unidad de control : ▨

+ buses internos ⟹

+ señales de control ⟶, ⟹ que van de la unidad de control a los componentes.

La ALU (Arithmetic/Logic Unit) realiza todos los cálculos del tipo $z = x$ op-alu y en donde op-alu es la operación que indique la unidad de control. Además indica las características del resultado en ZVSC

Control Unit

Circuito secuencial muy complejo que genera las señales de control. Estas hacen fluir los datos y direcciones por los buses que interconectan las unidades de almacenamiento y las de cálculo.

R-Banks (Register Banks)

Almacena los registros R0-R31. En S_1 y S_2 se indica el nº de los registros que aparecen constantemente por Rs_1 y Rs_2. Además si WPC = 1 se escribe en Rd en forma síncrona (la actualización se produce en el pulso de bajada).

Registros síncronos: PC y SR, mantienen el contador de programa y el registro de estado.

Registros asíncronos (latch): IR y AR, mantienen la instrucción en curso y la dirección que se debe colocar en el bus de direcciones.

Otros:

R-SEL: extrae de la instrucción los nºs de los registros que intervienen en una operación.

Y-SEL: elige el 2do operando de una instrucción, el que puede ser 0, 4 o lo que diga la instrucción (Rs_2 o imm) o un desplazamiento de 24 bits en caso de un salto.

Branch Unit: calcula la condición de salto durante instrucción del tipo b<cond> <label>

DBI (Data Bus Interface): interfaz con el bus de datos. Durante lecturas y escrituras, la memoria espera bytes y half-words en posiciones distintas de donde se encuentran en la palabra original. DBI desplaza los datos y extiende el signo si es necesario.

ABI (Address Bus Interface): genera los valores de A31-A2 y BE3-BE0 a partir de una dirección y el tipo de operación que le indique la unidad de Control.

Funcionamiento:
En cada ciclo del reloj. la unidad de control genera las señales de control para llevar los datos de los registros a los buses externos y a las unidades de cálculo.
Restricciones.

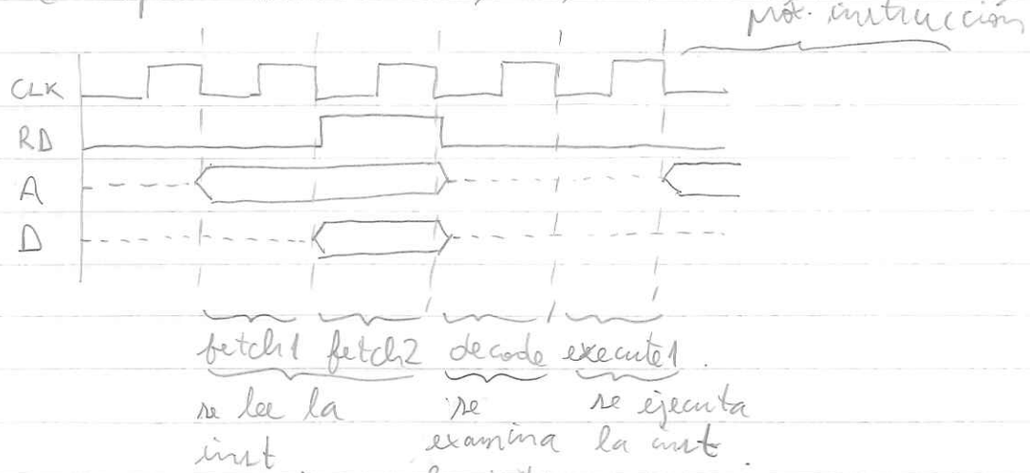
+ Las señales de control permanecen constantes durante todo un ciclo del reloj. Solo cambian en el pulso de bajada del reloj.

+ En un bus se puede colocar un y solo un dato en un ciclo del reloj.

+ La actualización del PC, SR y Rd ocurre solo en el pulso de bajada del reloj.

+ Cada unidad de cálculo puede realizar un y solo un cálculo en cada ciclo del reloj.

Etapas en la ejecución de una instrucción del tipo add R3, R4, R11



CPU 4

Para describir las operaciones que se llevan a cabo en cada ciclo se utilizan dos niveles de abstracción.

Transferencia entre registros	Señales de control
fetch1: AR ← PC	OP-Y-SEL ← @Ø
goto fetch2	OP-ALU ← @OR
	WR-AR, EN-A
	OP-ABI ← @W
fetch2: IR ← Mem^{W}[AR]	OP-DBI ← @LDW
if WAIT goto fetch2	SEL-D, WR-IR, EN-A, RD
else goto decode	OP-ABI ← @W
decode: PC ← PC ⊕ 4	OP-Y-SEL ← @4
goto execute1	OP-ALU ← @ADD
	WR-PC

Importante: en un mismo ciclo el orden en que se indican las transferencias o señales de control es irrelevante pues todas ocurren al mismo tiempo (en paralelo).

Las señales de control no especificadas permanecen en Ø. Las transferencias entre registros están restringidas por lo que permiten hacer las componentes, buses y señales de control.

Supongamos que la instrucción es add R4, -103, R11

execute1: R11 ← R4 ⊕ -103	SEL-REG, WR-Rd, WR-SR
[op ≡ add] goto fetch1	OP-Y-SEL ← @INST
	OP-ALU ← @ADD

CPU 5

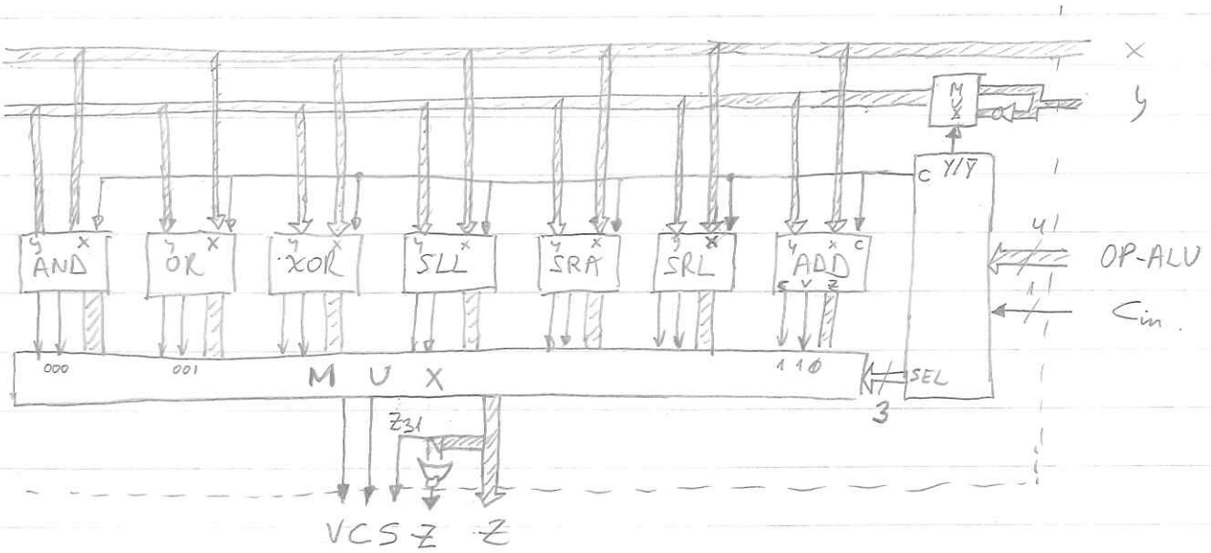
Supongamos que la instrucción es `stb R3, [R5+R0]`

execute 1	AR ← R5 ⊕ R0	SEL-REG, WR-AR, EN-A
[op ≡ stb]	goto execute 2	OP-Y-SEL ← @INST
		OP-ALU ← @ADD
		OP-ABI ← @W
execute 2	$Mem^{b}[AR] \leftarrow Trunc^{b}(R3)$	SEL-REG, RD-DEST
[op ≡ stb]	if WAIT goto execute 2	OP-Y-SEL ← @0
	else goto fetch 1	OP-ALU ← @OR
		OP-DBI ← @STB
		OP-ABI ← @B
		EN-D, EN-A, WR

Supongamos que la instrucción es `bg <label>`.

execute 1	if br? goto execute 2	nada
[op ≡ bg]	else fetch 1	
execute 2	$PC \leftarrow PC \oplus Ext^{S}(IR[23\text{-}0])$	OP-Y-SEL ← @DISP
[op ≡ bg]	goto fetch 1	OP ← ALU ← @ADD
		WR-PC

Implementación de la ALU



En cada ciclo del reloj, la ALU realiza en paralelo las 7 operaciones que implementa. Un multiplexor se encarga de seleccionar la operación que indique la Unidad de Control.

La unidad ADD calcula $V = \begin{cases} 1 & \text{si } x_{31} = y_{31} \neq z_{31} \\ \emptyset & \text{sino} \end{cases}$.

C = último carry

El resto de las unidades $V = \emptyset$

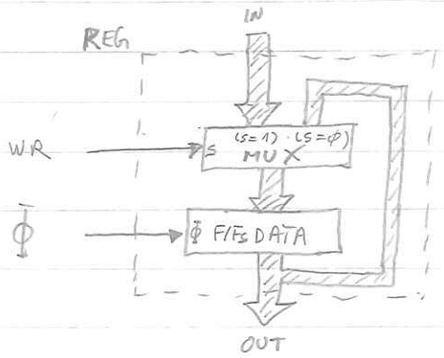
$C = C_{in}$

Ejercicio: Completar la tabla de verdad para el circuito combinacional:

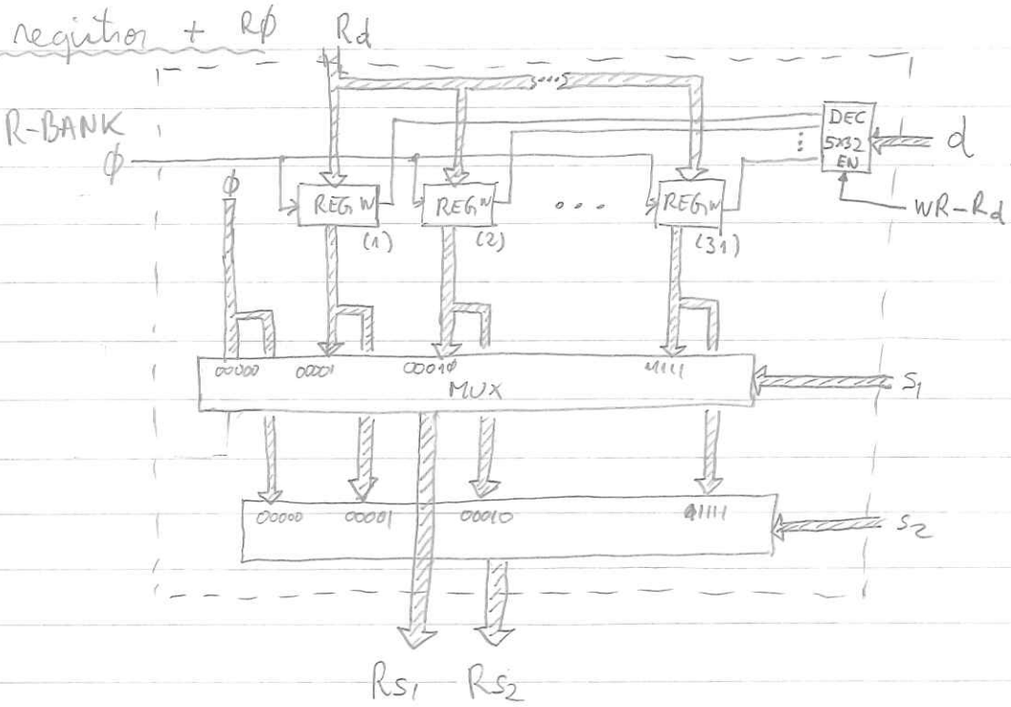
	OP-ALU	Cin	C	SEL	Y/Ȳ	calcula
	⋮					
(ADD)	0000	X	$\emptyset$	11$\emptyset$	1	$x \oplus y$
(ADDX)	$\emptyset\emptyset\emptyset$1	$\emptyset$	$\emptyset$	11$\emptyset$	1	$x \oplus y \oplus c$
		1	1	11$\emptyset$	1	
(SUB)	$\emptyset\emptyset$1$\emptyset$	X	1	11$\emptyset$	$\emptyset$	$x \oplus \sim y \oplus 1$
	⋮			⋮		

El banco de registros:

1 registro síncrono:

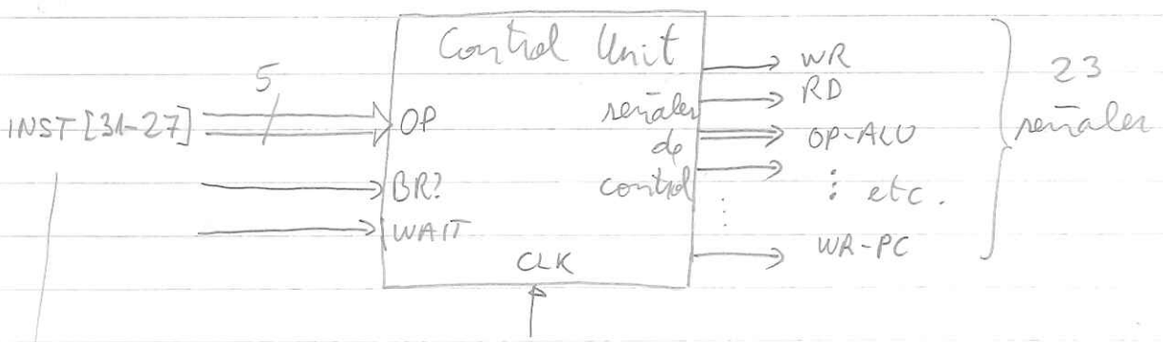


31 registros + Rφ



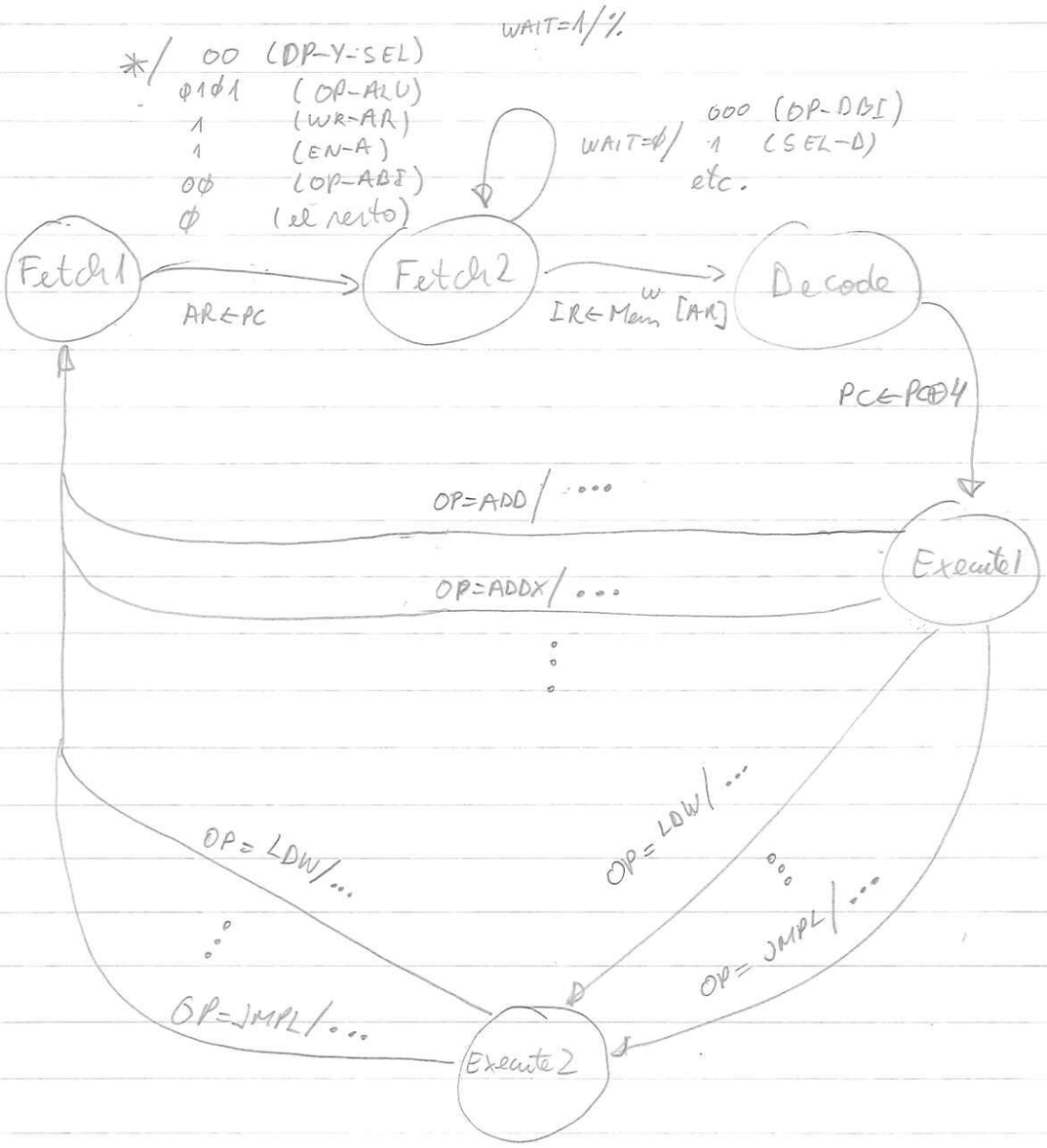
La unidad de control

Corresponde a un circuito secuencial complejo:



Dado que M32 no tiene más de 32 instrucciones sólo se necesitan 5 bits para codificar todas las instrucciones.

El diagrama de estados es:



En un ciclo ocurren las siguientes acciones:

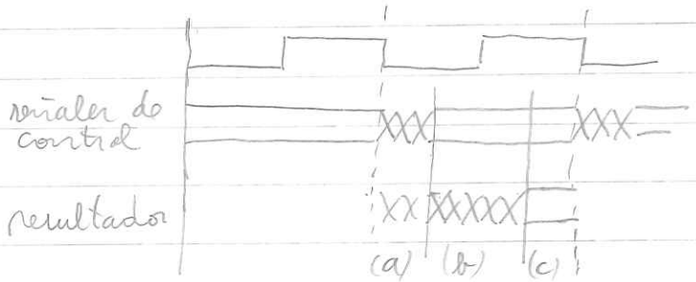
(a) la unidad de control calcula las señales de control para dirigir los datos hacia las unidades de cálculo.

(b) las unidades de cálculo efectúan sus cálculos

(c) los resultados son almacenados en algún registro.

Cada una de estas acciones toma un tiempo de retardo. El período del reloj debe ser fijado de modo que se alcancen a realizar en un ciclo

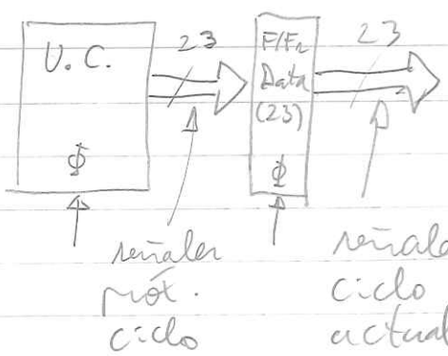
del reloj:



La evolución de los µ-P está dirigida por lograr mayor rapidez al mismo precio. Para lograr mayor rapidez se lucha por:

+ Hacer que los transistores reaccionen más rápido. ⇒ disminuye el período del reloj.
+ Mejorar la tasa de instrucciones ejecutadas por ciclo del reloj. (M32: 1 por c/4 o 5 ciclos).
+ Disminuir el tiempo de (a), (b) o (c).

Una forma de lograr esto último es: en vez de calcular las señales de control en este ciclo, calcularlas en el ciclo precedente. Dicho de otra forma: en cada ciclo se calculan las señales de control que se usarán en el próximo ciclo. Estas señales se dirigen a un registro sincrono que se actualiza en el pulso de bajada del reloj.



De esta forma se suprime la componente (a) del período del reloj. Notese que esta optimización no sería posible aplicarla si no existiera el ciclo Decode, pues la instrucción que fijará las señales de control que se aplicarán en el ciclo Execute