Electronica digital 2

Introducción al procesador

Ferney Alberto Beltrán Molina

Agosto 2019

Contacto

Nombre:	Ferney Alberto Beltrán Molina, Ing, MSc, PhD(c)
Email:	fabeltranm@unal.edu.co
oficina:	

Contenido

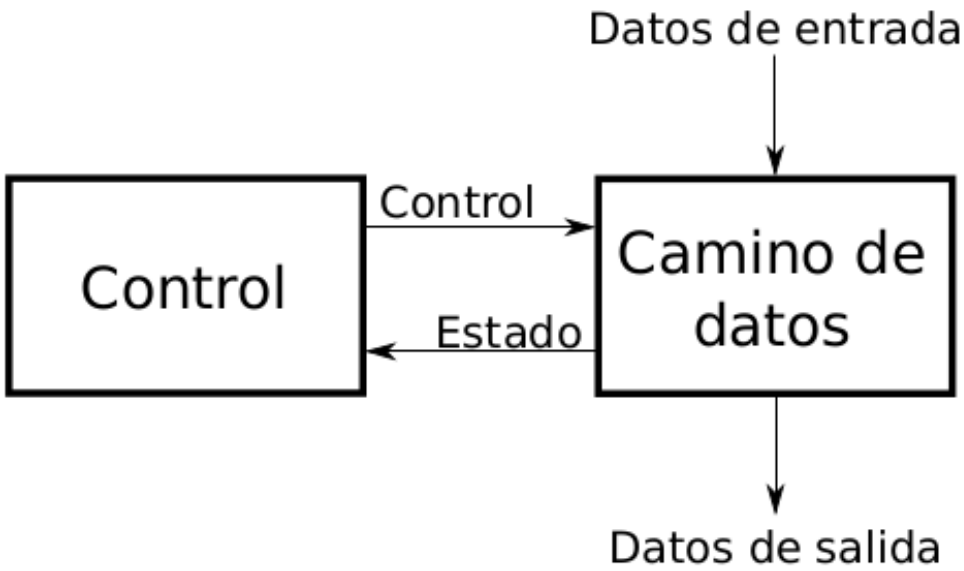
Índice

Antes..

En la clase anterior vimos:

- Se presenta un ejemplo de Contador y se reviso el divisor
- Se establece el proceso realizado:
  - 1. Elabora un diagrama de flujo que describa la funcionalidad deseada.
  - 2. Identificar los componentes del DataPath.
  - 3. Identificar las señales necesarias para controlar el Datapath y la interconexión.
  - 4. Especificar la unidad de control (FSM) utilizando diagramas de estado.
  - 5. Simulación y pruebas.

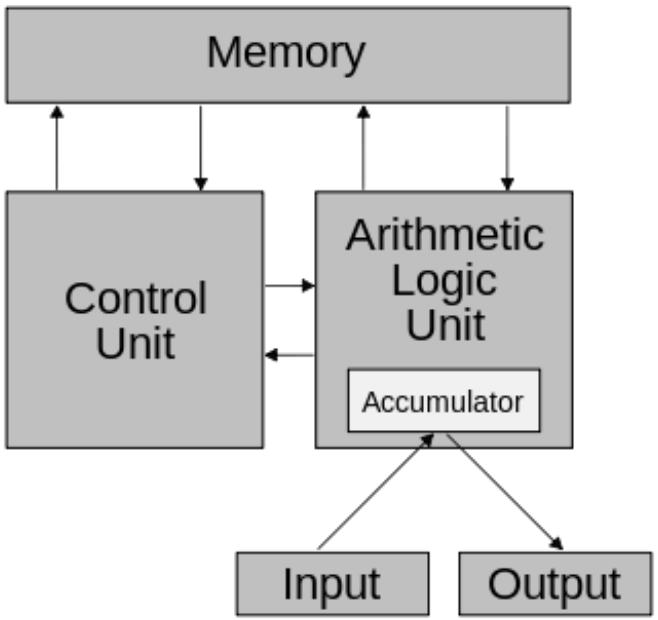
Recordando


Datos de entrada
Control
Control
Camino de datos
Estado
Datos de salida


CASIO

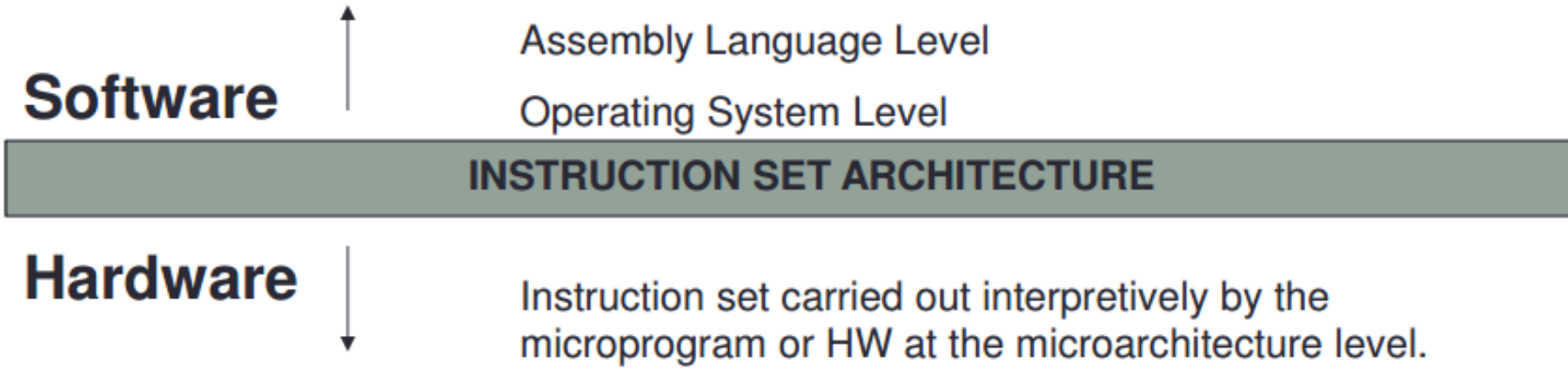
Índice

Hardware Software Interface



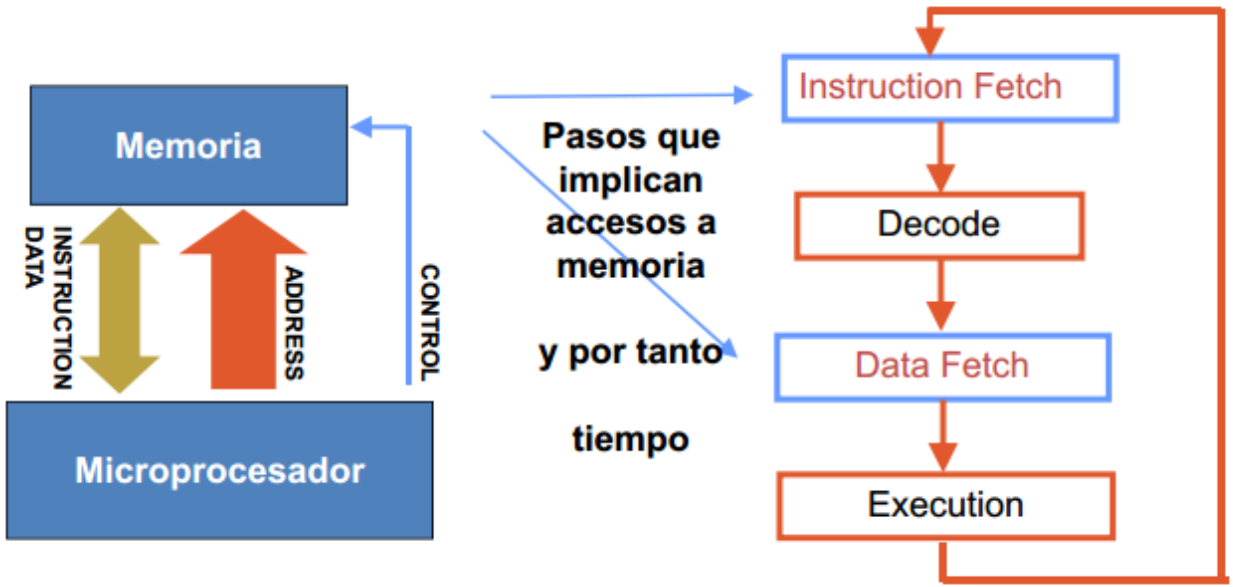
Hardware Software Interface

La frontera entre el hardware y el software



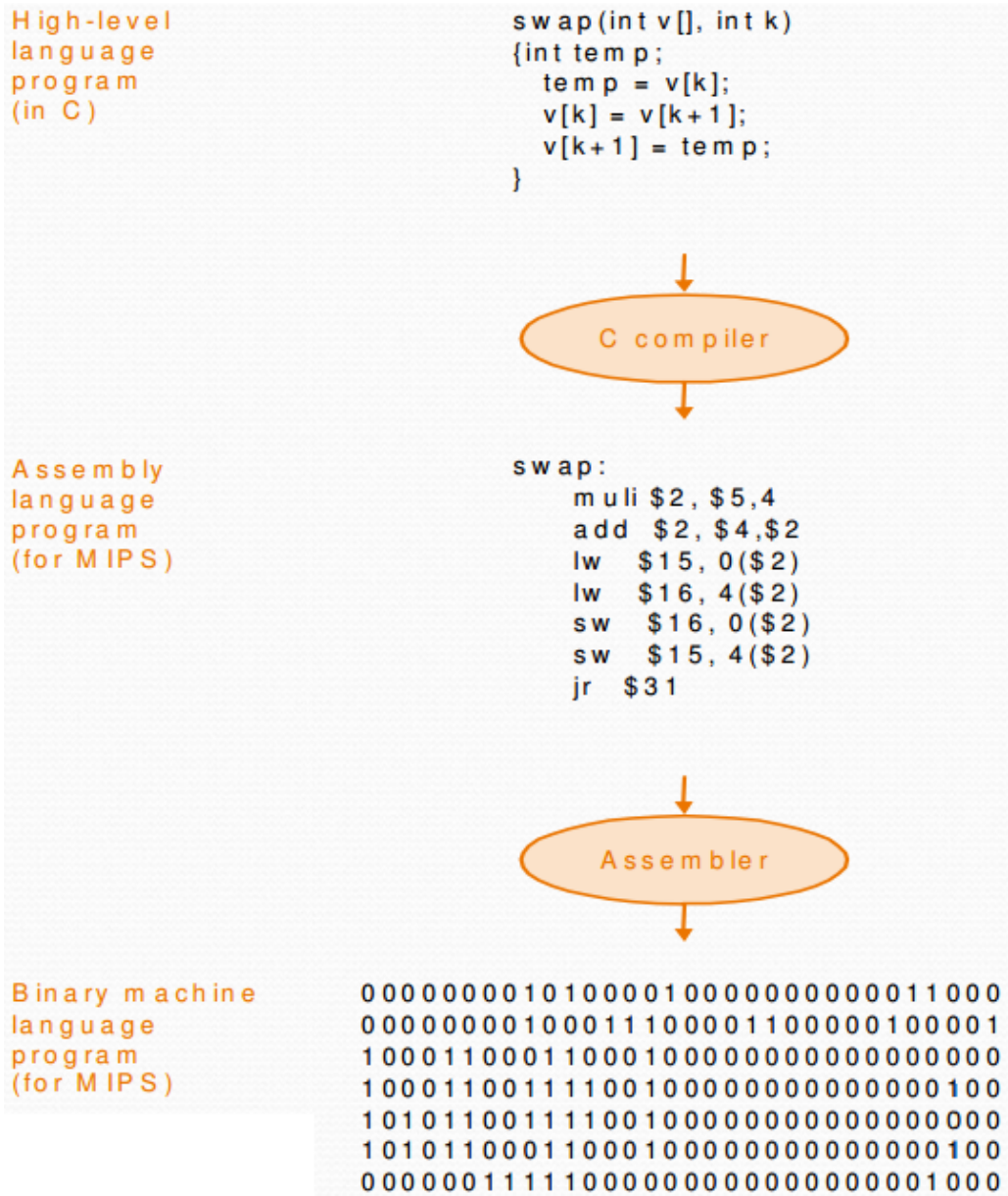
Cada instrucción es directamente ejecutada por el hardware

Ciclo de instrucción



- Buscar la instrucción en la memoria principal
- Decodificar la instrucción
- Ejecutar la instrucción
- Almacenar o guardar resultados

Abstracción



Diseño el set de instrucciones (ISA)

- Simplicidad favorece la regularidad
- Cuanto más pequeños, más rápido
- Hacer lo común, lo más rápido
- Un buen diseño exige buenos compromisos

Set de instrucciones para arquitecturas basadas en pilas.
Set de instrucciones para arquitecturas basadas en registros.

cómo se representa

- Con un formato binario. El hardware solo entiende bits
- Los objetos físicos son bits, bytes, palabras (words).
- Tamaño típico de palabra: 4 u 8 bytes (32 o 64 bits).

- Se identifica por un opcode (código de operación)
  add $so,$s1, $s2
- Requiere de 0 a 3 operandos.
  identificador de la zona donde estan almacenados
  memoria, registros, stack

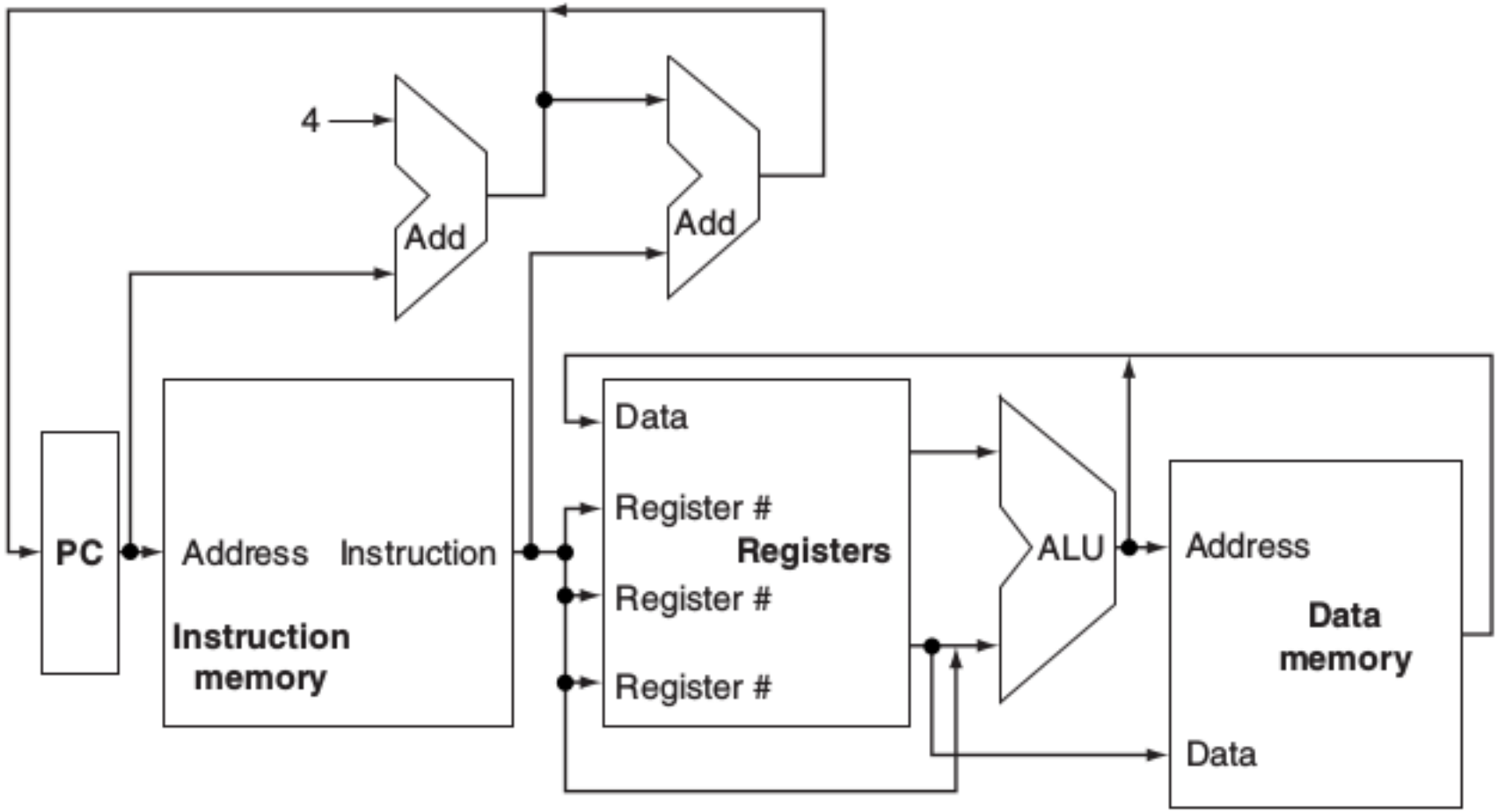
El procesador

Sigue las instrucciones del programa al pie de la letra. Suma y compara números, ordena activarse a los dispositivos de I/O, etc.

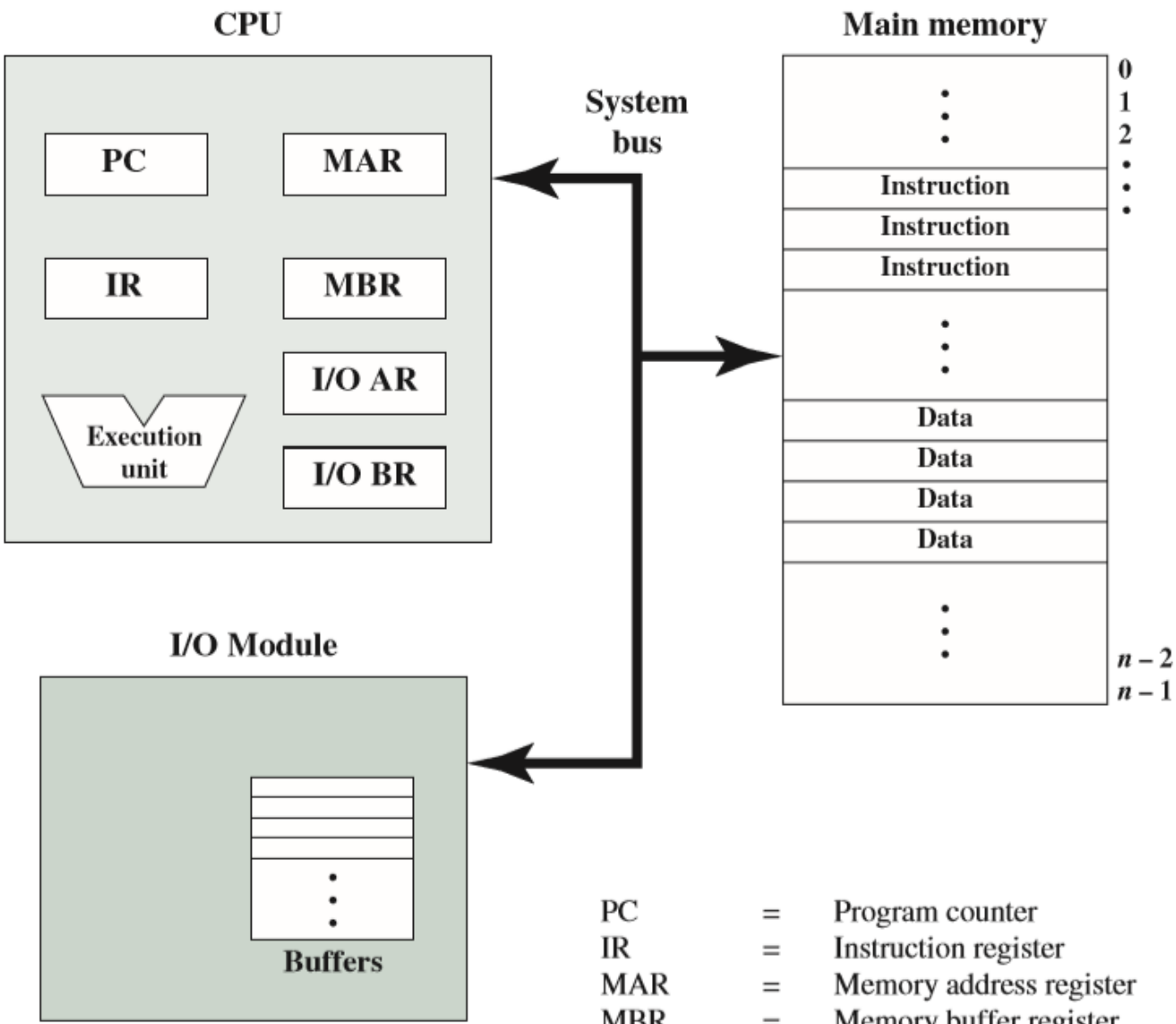
El procesador consta de dos componentes:

- El datapath. Ejecuta operaciones aritméticas y lógicas.
- El control. Ordena al datapath, memoria y dispositivos de I/O lo que hay que hacer de acuerdo al program

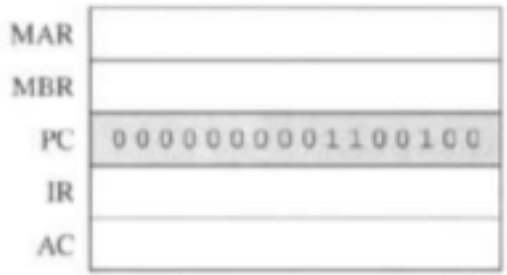
Procesador básico



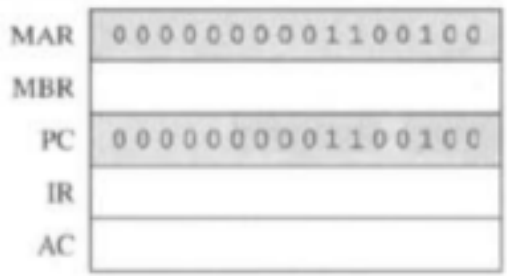
Ciclo de instrucción



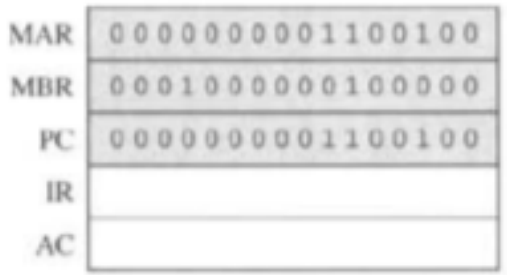
Ciclo de instrucción



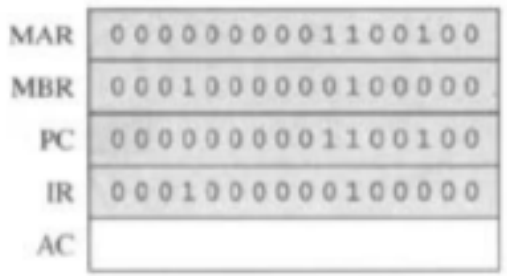
(a) Comienzo



(b) Primer paso



(c) Segundo paso



(d) Tercer paso

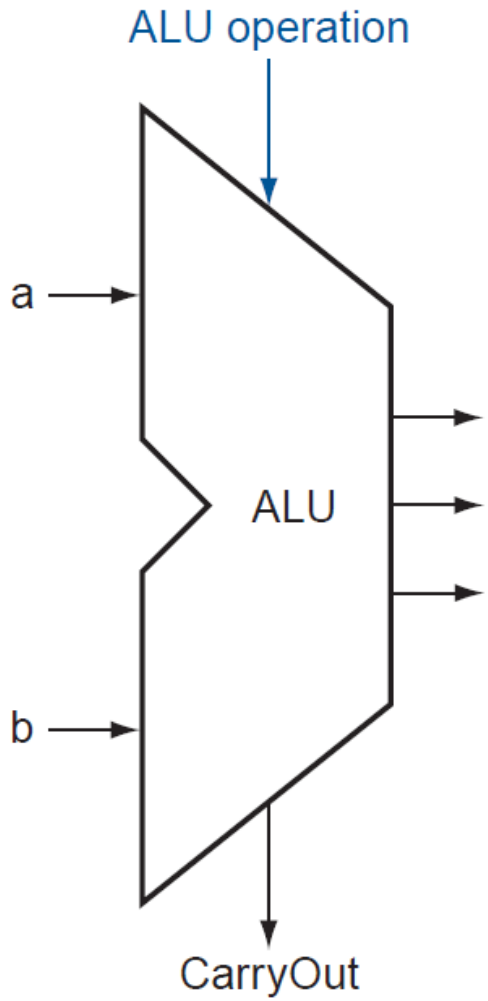
- t1: Memory address Register (MAR) $\longleftarrow$ PC
- t2: Memory Buffer Register (MBR) $\longleftarrow$ Memoria
  t2: Program Counter (PC) $\longleftarrow$ PC+1
- t3: Instruction Register (IR) $\longleftarrow$ MBR

Datapath

Unidad aritmético-lógica
La cual se encarga de realizar las operaciones que requiera el algoritmo.
Lógicamente las arquitecturas no tienen implemenentadas todas las posibles funciones matemáticas o funciones aritméticas (instrucciones)

Banco de registros
Mantiene almacenada la información o los datos
Registros de acceso a memoria, registros programados, registro especiales etc.

Datapath (Componentes básicos)

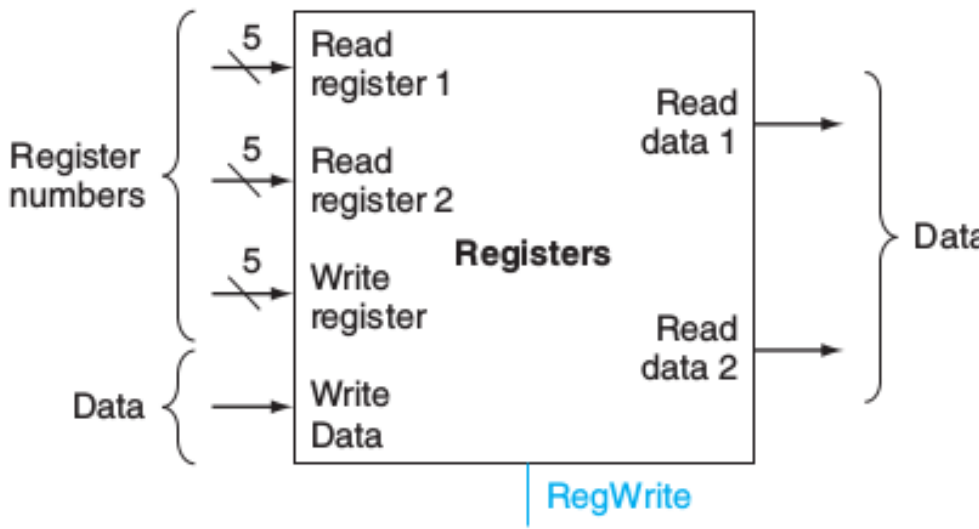


code	operation
0	T
1	N
2	$T + N$
3	$T \texttt{and} N$
4	$T \texttt{or} N$
5	$T \texttt{xor} N$
6	$\sim T$
7	$N = T$
8	$N < T$
9	$N \texttt{rshift} T$
10	$T - 1$
11	R
12	$[T]$
13	$N \texttt{lshift} T$
14	depth
15	$N \texttt{u<} T$

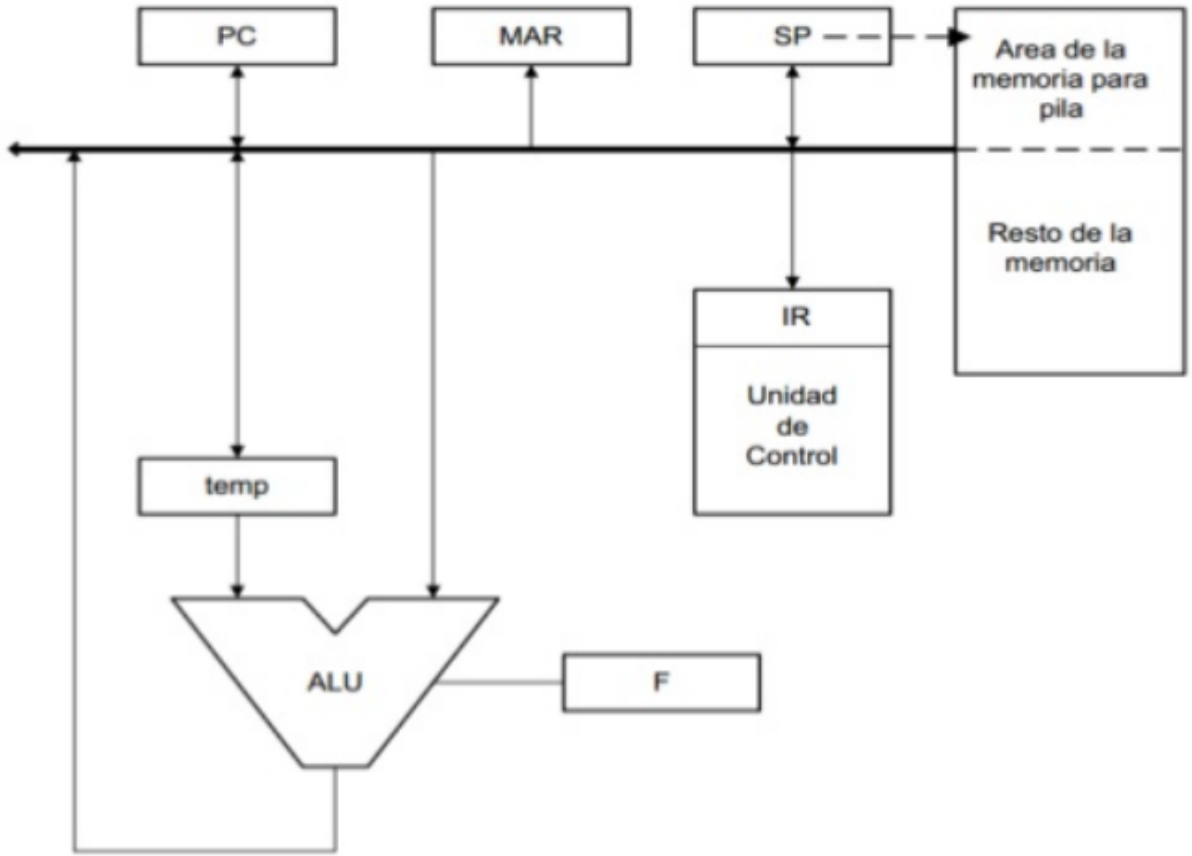
Banco de Registros

El banco de registros (register file) es un conjunto de registros para guardar y leer datos.

1. Cada registro es un vector de flip-flops D.
2. Para leer un registro:
   - 2.1 Entrada: número de registro.
   - 2.2 Salida: dato contenido en el registro.
3. Para escribir un registro:
   - 3.1 Entrada: número de registro, dato y una señal de reloj para controlar la escritura.



Datapath (stack)



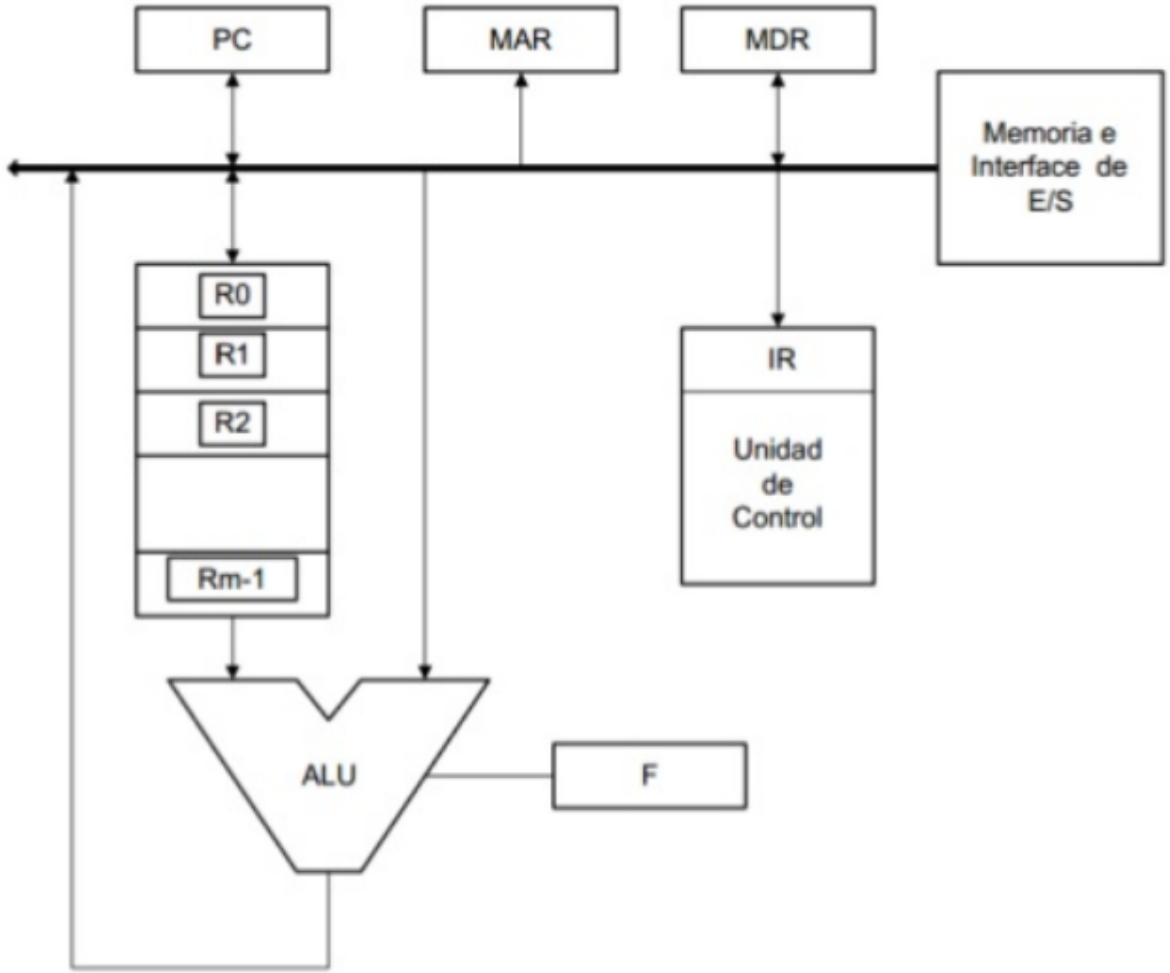
SP: Puntero de la Pila: registro con la dirección de la ultima palabra insertada en la pila **TOS, Top od Stack**
en funcion de Sp se obtiene el próximo registro **NOS Next of Stack**

Direccionamiento

Implicito

- El opcode implica la dirección de los operandos.
- Ejemplo en una máquina de pila (stack) **add**
- La instrucción saca (pop) dos valores de la pila, hace la suma y deja (push) el resultado en la pila

DataPath Acumulador



Banco de registros de propósito general ($R_o \ldots R_{m-1}$)

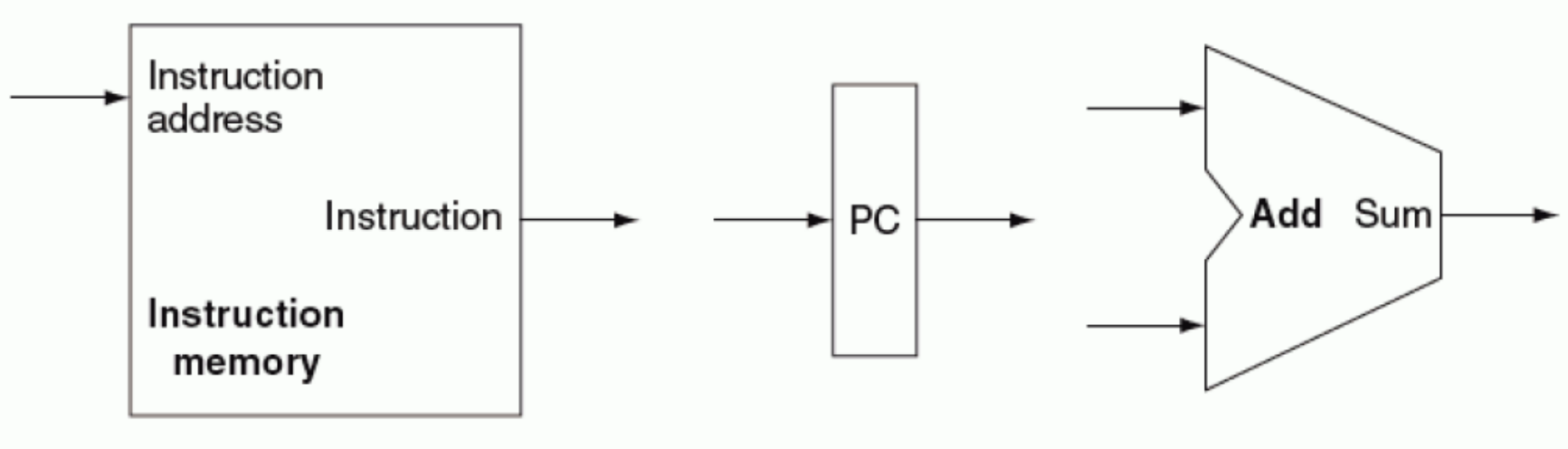
Maquinas de dos (Lectura destructiva) o tres operandos

Direccionamiento

Explícito

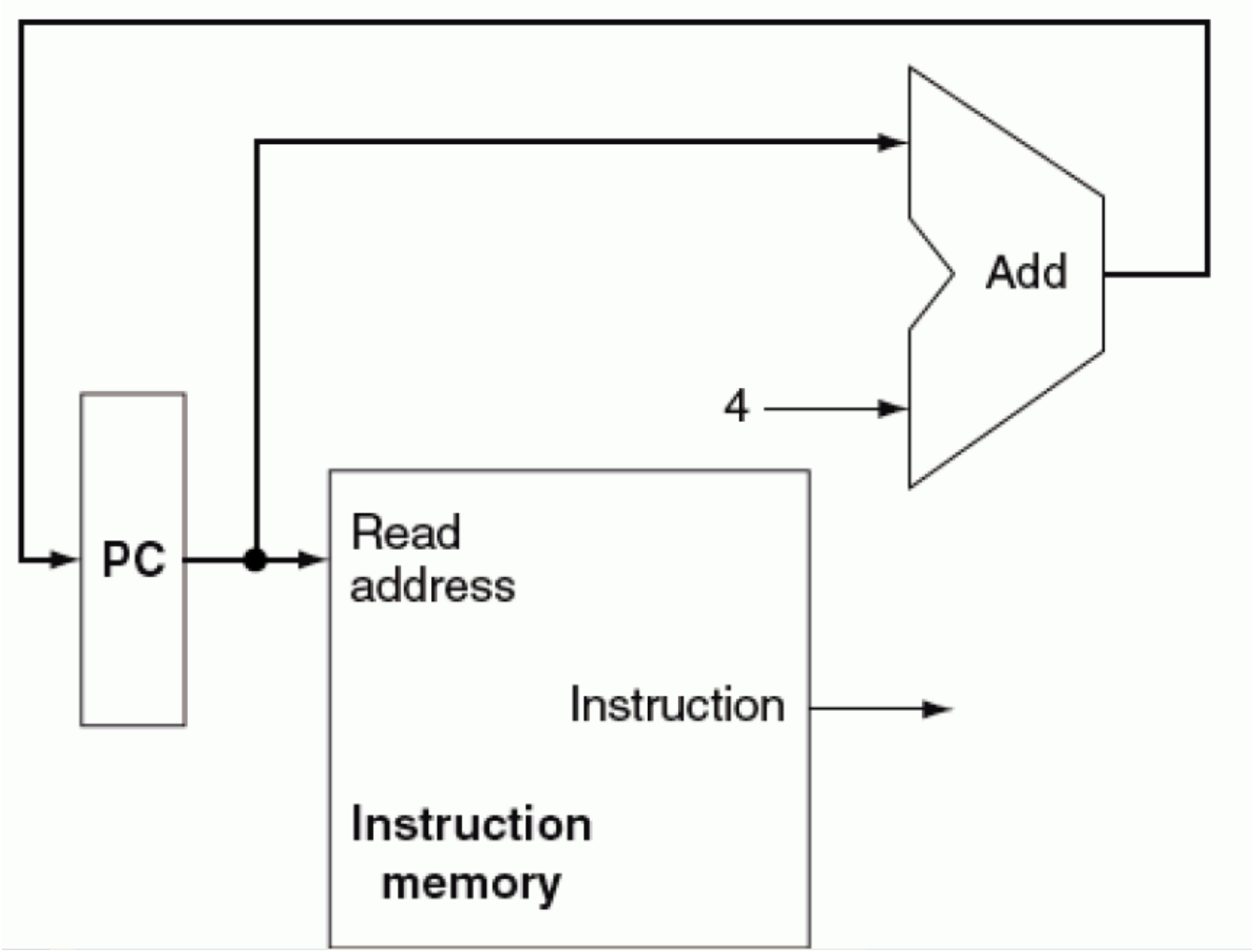
- Las direcciones vienen en los operandos.
- Ejemplo de MIPS:
  add $so,$s1, $s2
- Dos operandos fuentes: s1 y s2
- Un operando destino: s0
- s0 = s1 + s2

Componentes básicos adicionales



Una memoria para guardar y leer instrucciones
Un registro, llamado PC (contador de programa), para guardar la dirección de la instrucción actual.
Un sumador para incrementar el PC.

Componentes básicos adicionales

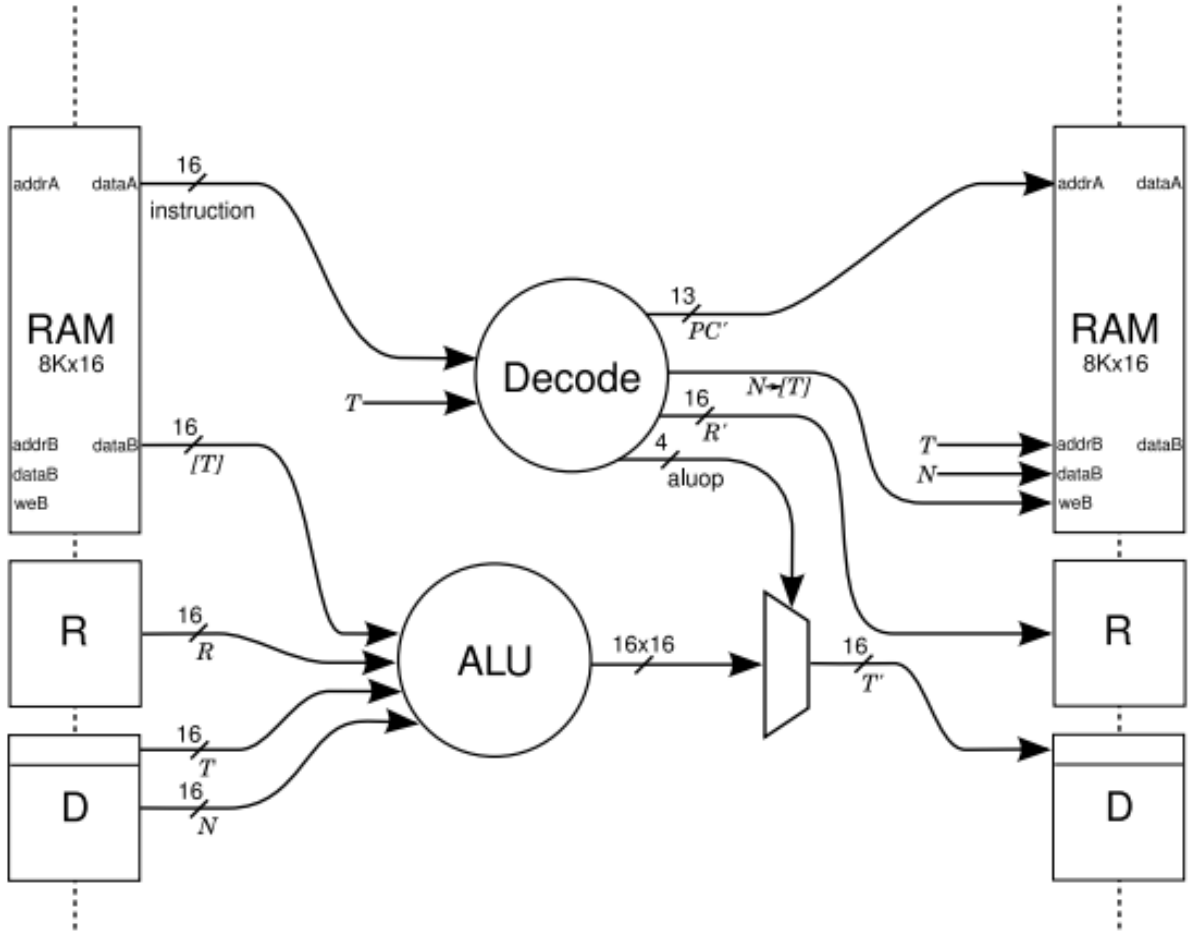


Obtener la instrucción de la memoria.

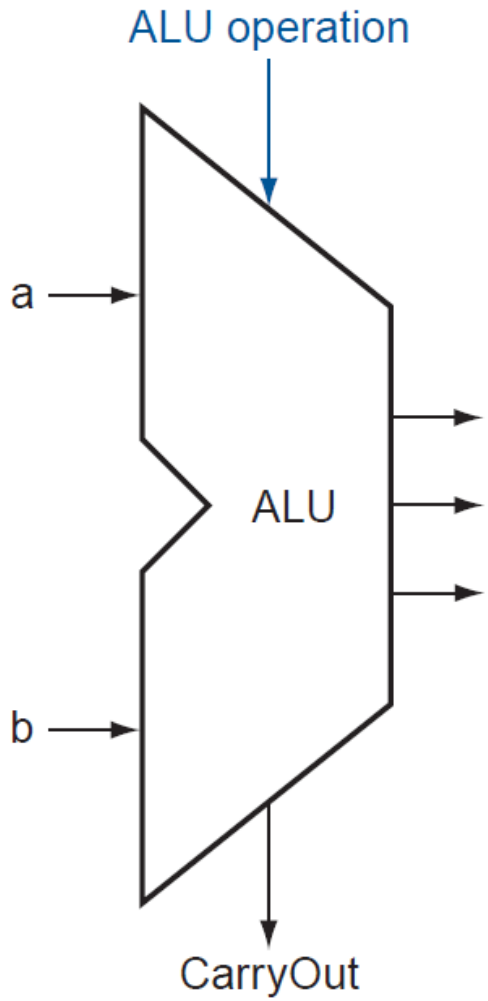
Incrementar el PC para preparar la ejecución de la instrucción siguiente.

Índice

Datapath J1 (Componentes básicos)



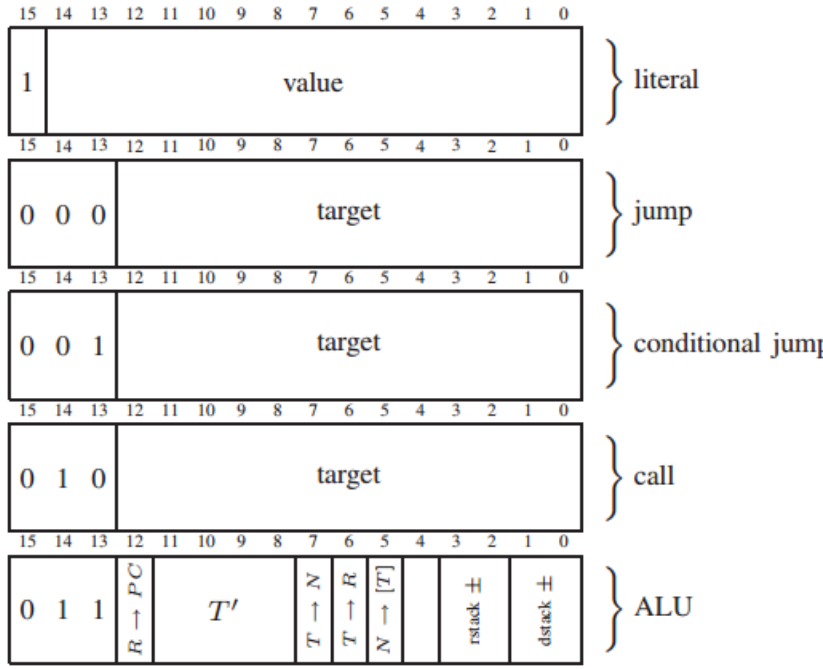
Datapath (Componentes básicos)



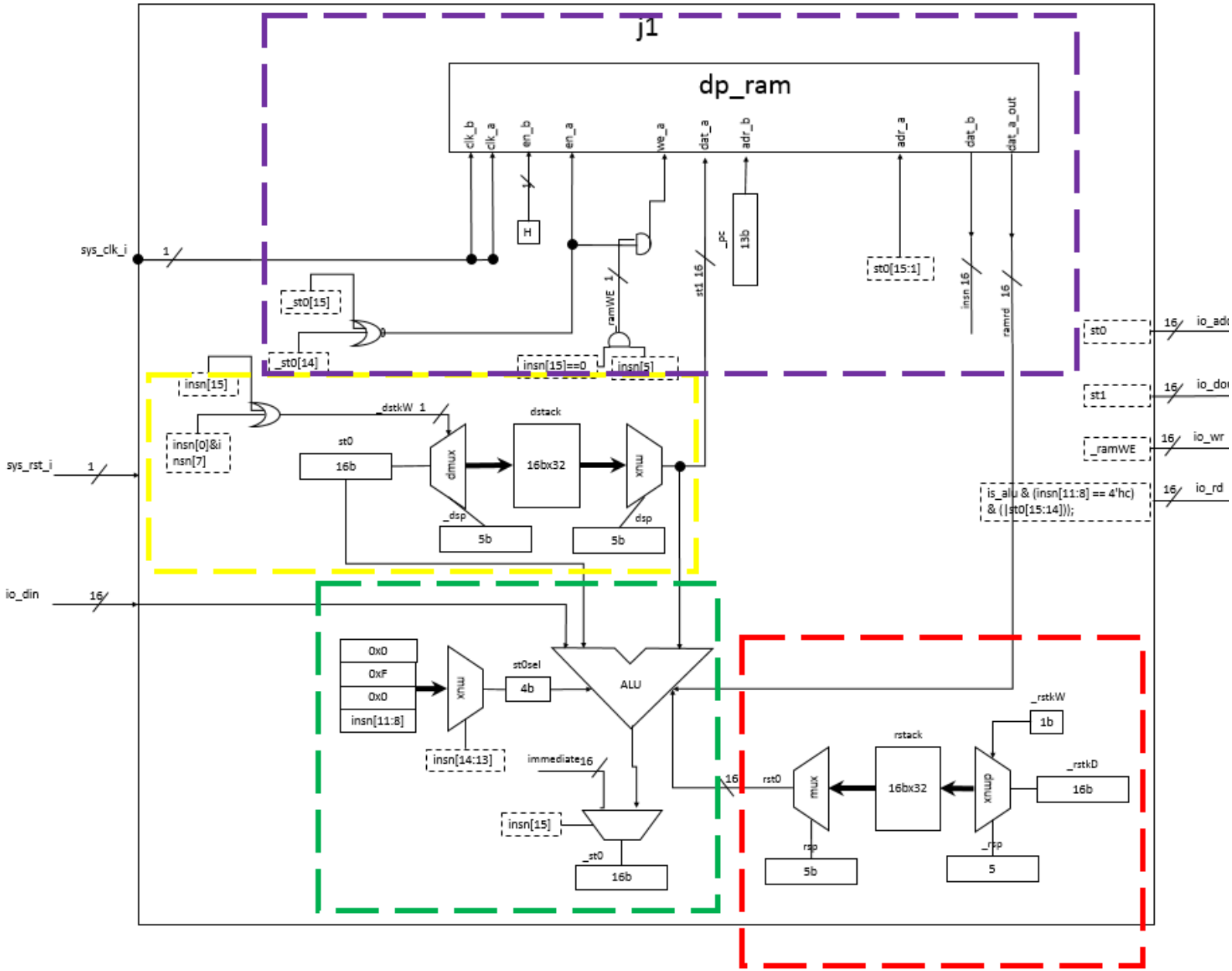
code	operation
0	T
1	N
2	$T + N$
3	$T \texttt{and} N$
4	$T \texttt{or} N$
5	$T \texttt{xor} N$
6	$\sim T$
7	$N = T$
8	$N < T$
9	$N \texttt{rshift} T$
10	$T - 1$
11	R
12	$[T]$
13	$N \texttt{lshift} T$
14	depth
15	$N \texttt{u<} T$

J1

field	width	action
T'	4	ALU op, replaces T, see table II
$T \rightarrow N$	1	copy T to N
$R \rightarrow PC$	1	copy R to the PC
$T \rightarrow R$	1	copy T to R
dstack $\pm$	2	signed increment data stack
rstack $\pm$	2	signed increment return stack
$N \rightarrow [T]$	1	RAM write

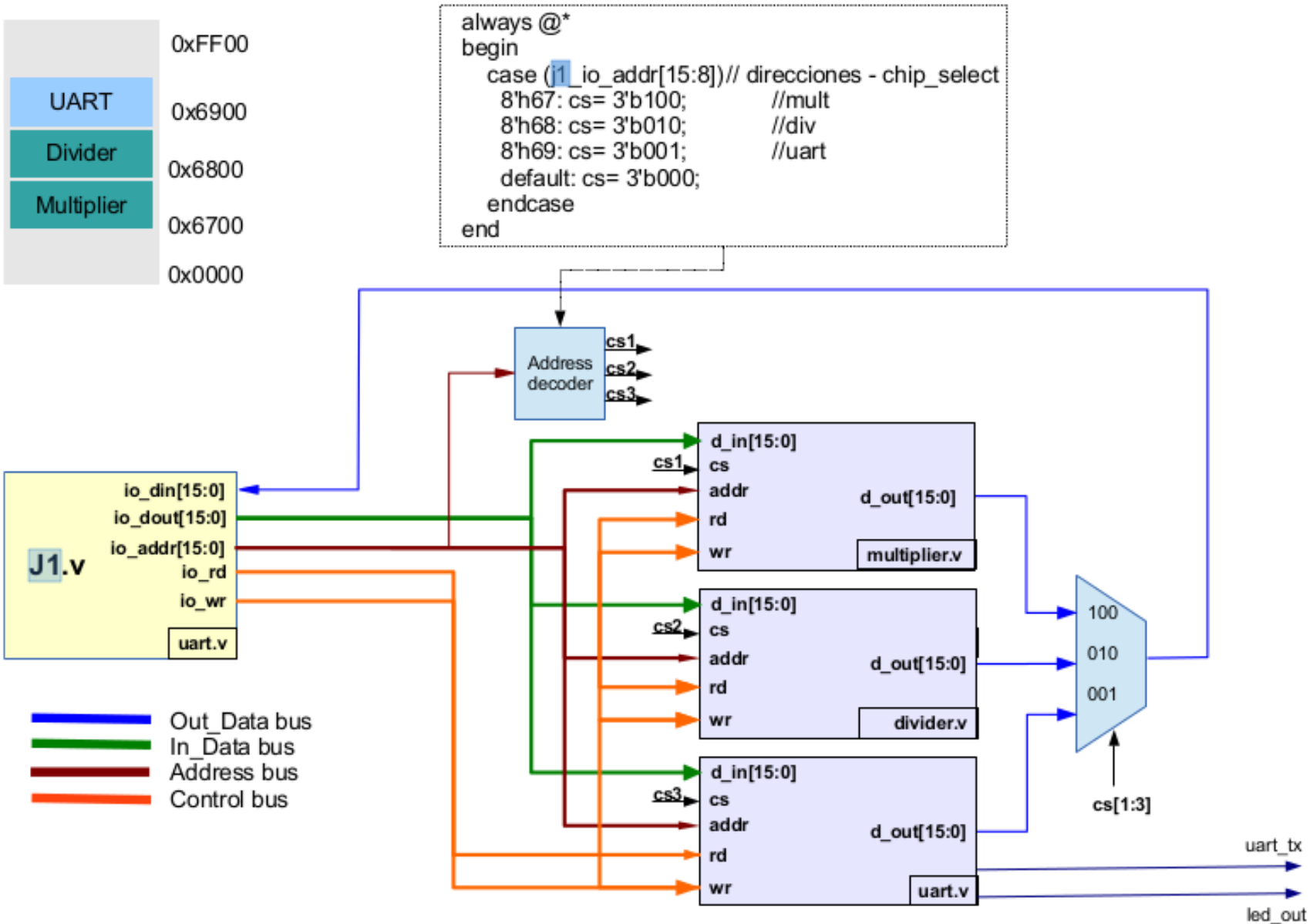


J1


j1
dp_ram
clk_b
clk_a
en_b
en_a
we_a
dat_a
adr_b
adr_a
dat_b
dat_a_out
H
sys_clk_i
1
_st0[15]
_st0[14]
ramWE 1
insn[15]==0
insn[5]
st1 16
_pc
13b
st0[15:1]
insn 16
ramrd 16
st0
st1
_ramWE
is_alu & (insn[11:8] == 4'hc) & (|st0[15:14]));
io_adr
io_dout
io_wr
io_rd
insn[15]
insn[0]&insn[7]
sys_rst_i
_dstkW 1
dstack
st0
16b
dmux
16bx32
mux
_dsp
dsp
5b
5b
io_din
16
0x0
0xF
0x0
insn[11:8]
mux
st0sel
4b
ALU
insn[14:13]
immediate 16
insn[15]
_st0
16b
_rstkW
1b
rstack
rst0
mux
16bx32
dmux
_rstkD
16b
_rsp
5b
_rsp
5

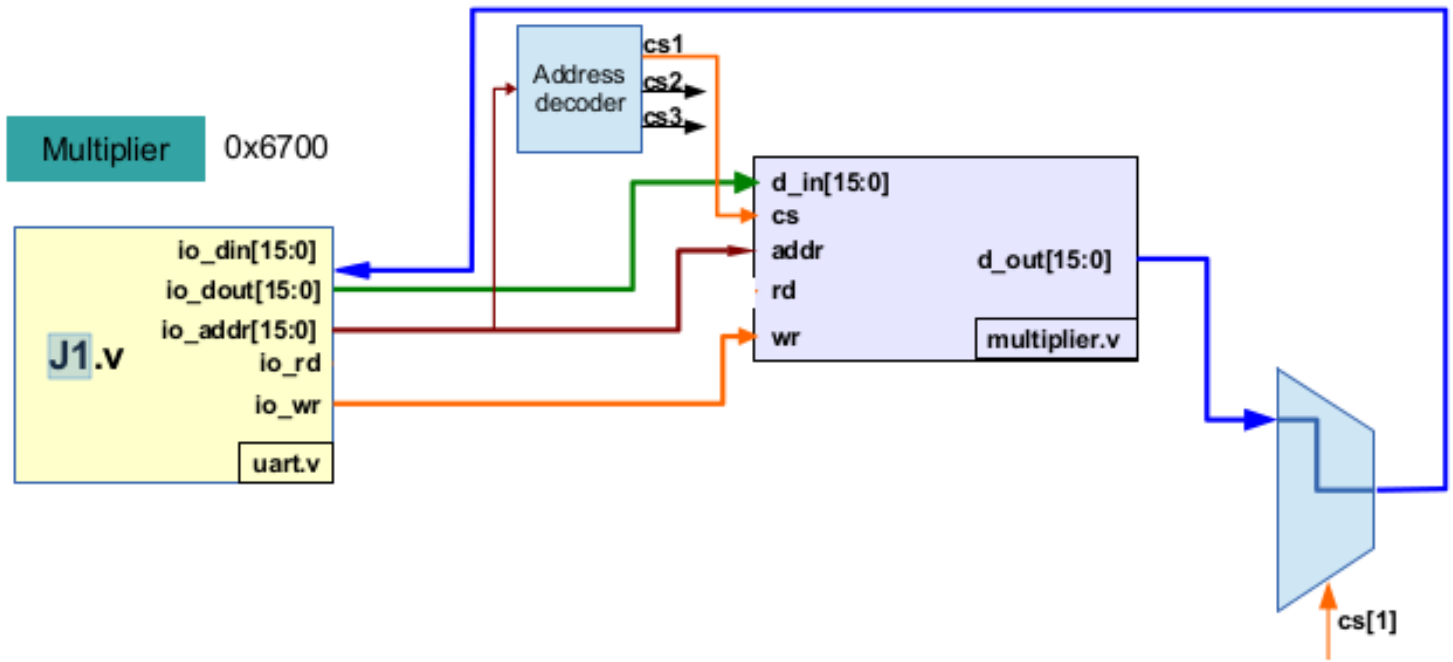
Índice

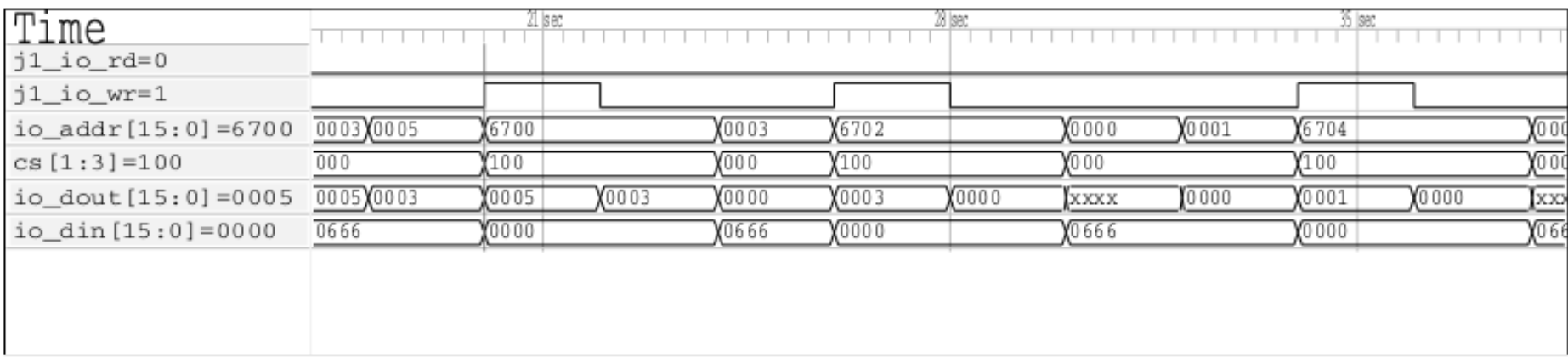
J1 CPU


0xFF00
UART
0x6900
Divider
0x6800
Multiplier
0x6700
0x0000
always @*
begin
case (j1_io_addr[15:8])// direcciones - chip_select
8'h67: cs= 3'b100; //mult
8'h68: cs= 3'b010; //div
8'h69: cs= 3'b001; //uart
default: cs= 3'b000;
endcase
end
Address decoder
cs1
cs2
cs3
J1.v
io_din[15:0]
io_dout[15:0]
io_addr[15:0]
io_rd
io_wr
uart.v
d_in[15:0]
cs
addr
rd
wr
d_out[15:0]
multiplier.v
divider.v
100
010
001
cs[1:3]
Out_Data bus
In_Data bus
Address bus
Control bus
uart_tx
led_out

J1 CPU escritura

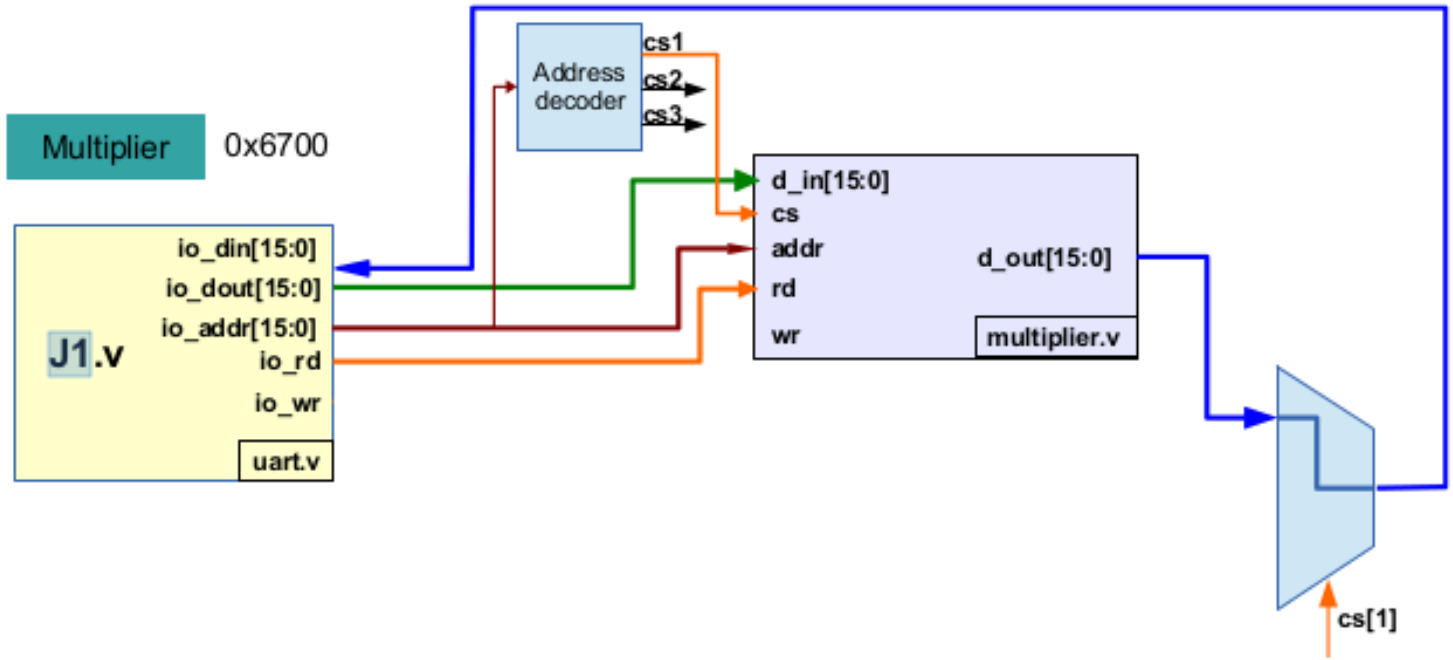
Escribir los datos 0x0005 en la dirección 0x6700, 0x0003 en la 0x6702 y 0x0001 en la 0x6704

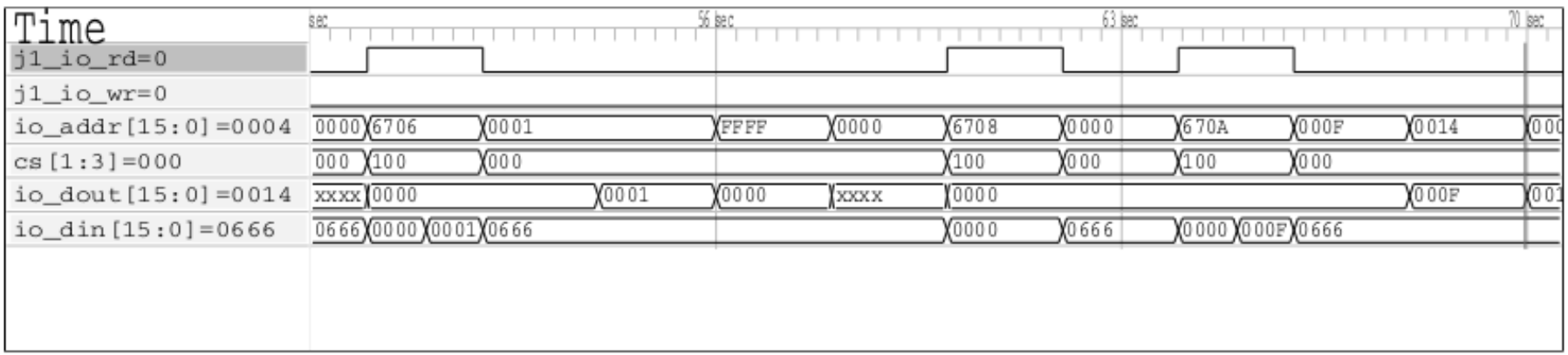




J1 CPU leer

Leer los datos de las direcciones 0x6706, 0x6708 y la 0x670A





J1 CPU leer

Almacenamiento por bytes

0x6700 **0005**
0x6702 **00003**
0x6704 **0001**
0x6706 **0001**
0x6708 **0000**
0x670A **000F**

0x6701 **00**	0x6700 **05**
0x6703 **00**	0x6702 **03**
0x6705 **00**	0x6704 **01**
0x6707 **00**	0x6706 **01**
0x6709 **00**	0x6708 **00**
0x670B **00**	0x670A **0F**

¿Cómo se almacena si el bus es de 32 bits?
¿Cuántos bytes se reservan?

Mapa de memoria Multiplicador

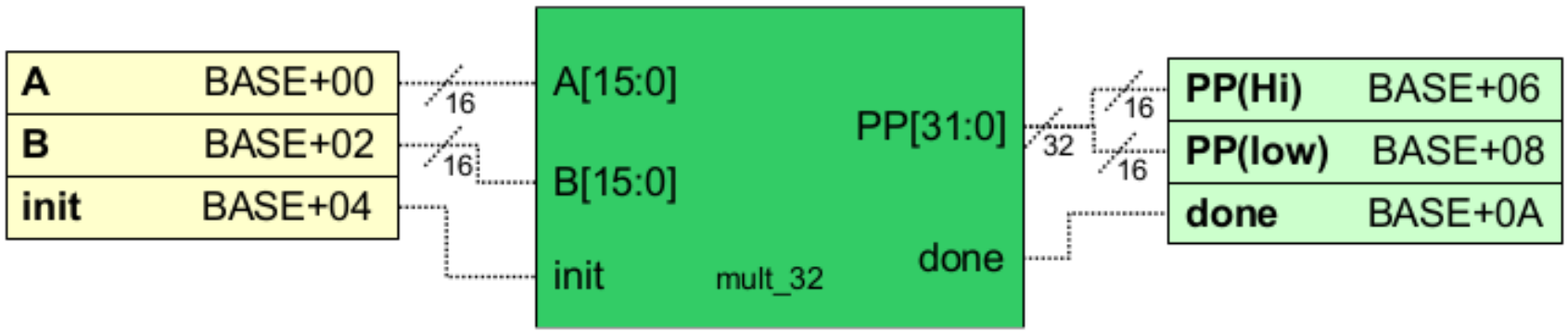
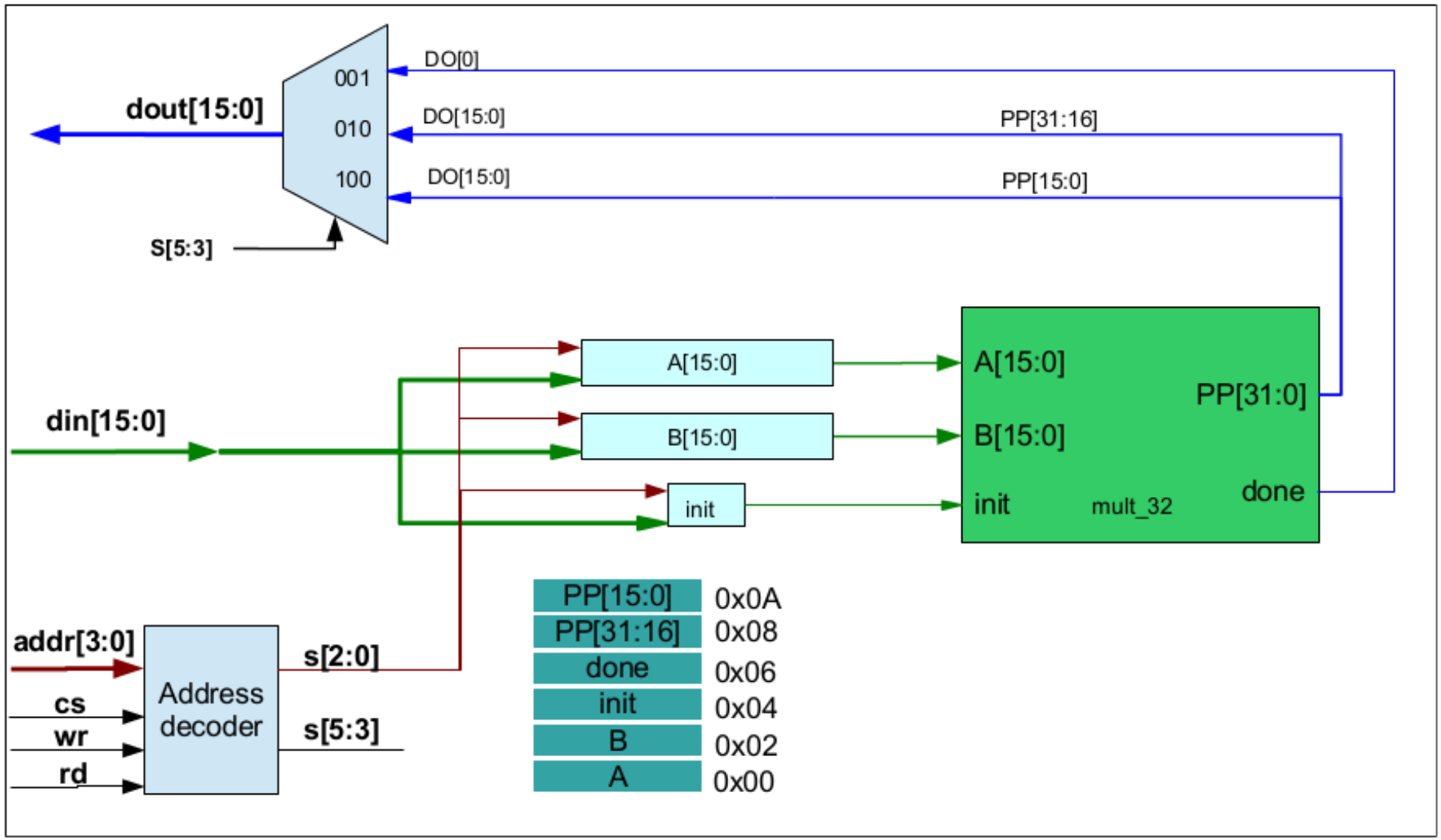
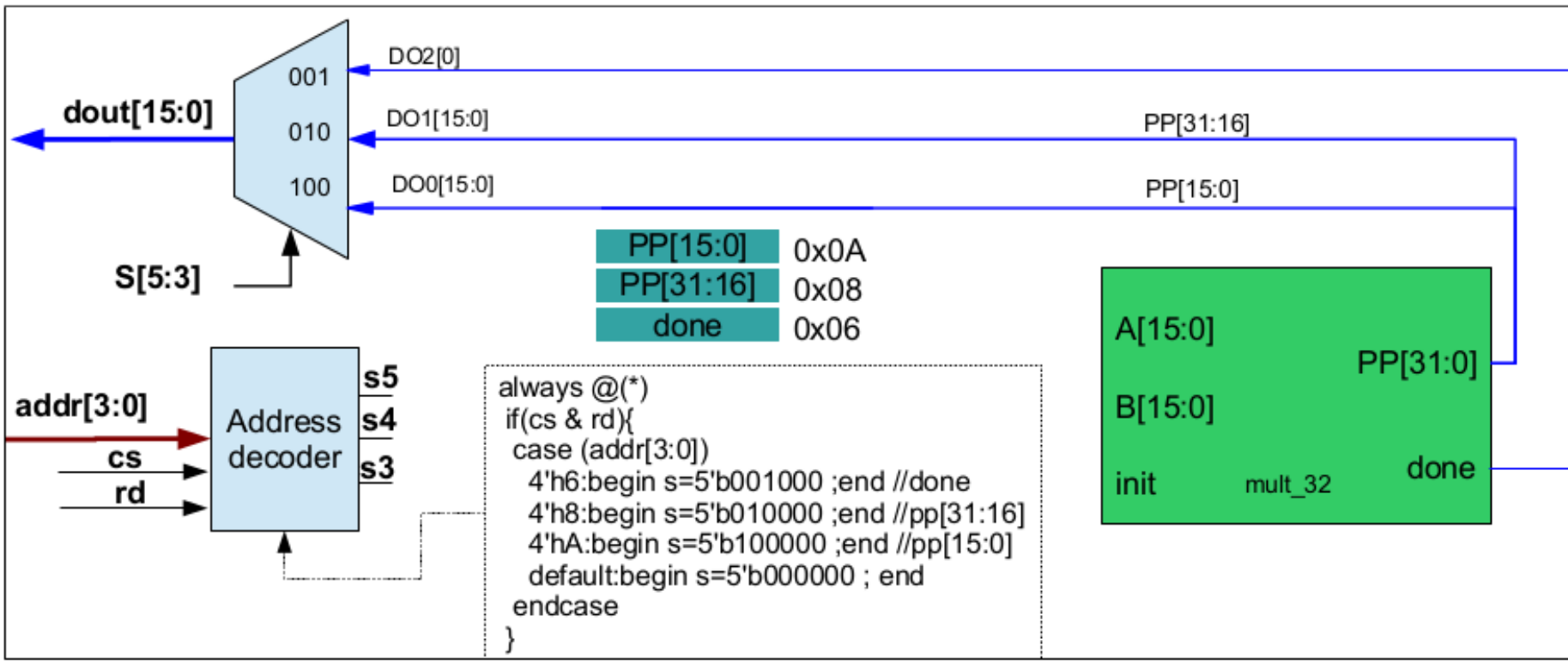


Diagrama de bloques Multiplicador



Lectura


001
010
100
dout[15:0]
DO2[0]
DO1[15:0]
DO0[15:0]
PP[31:16]
PP[15:0]
S[5:3]
PP[15:0] 0x0A
PP[31:16] 0x08
done 0x06
addr[3:0]
cs
rd
Address decoder
s5
s4
s3
always @(*)
if(cs & rd){
case (addr[3:0])
4'h6:begin s=5'b001000 ;end //done
4'h8:begin s=5'b010000 ;end //pp[31:16]
4'hA:begin s=5'b100000 ;end //pp[15:0]
default:begin s=5'b000000 ; end
endcase
}
A[15:0]
B[15:0]
init
mult_32
PP[31:0]
done

Escritura

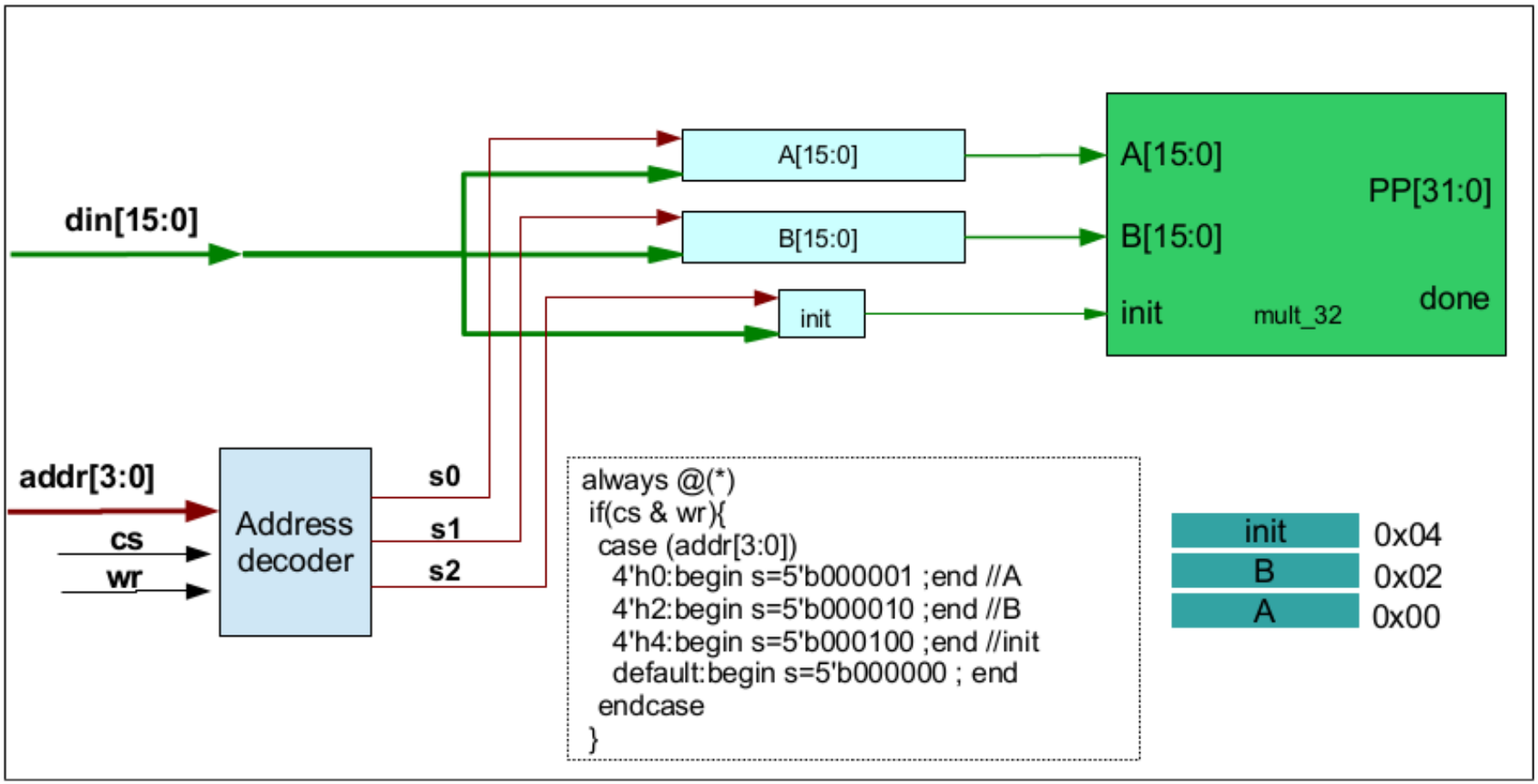
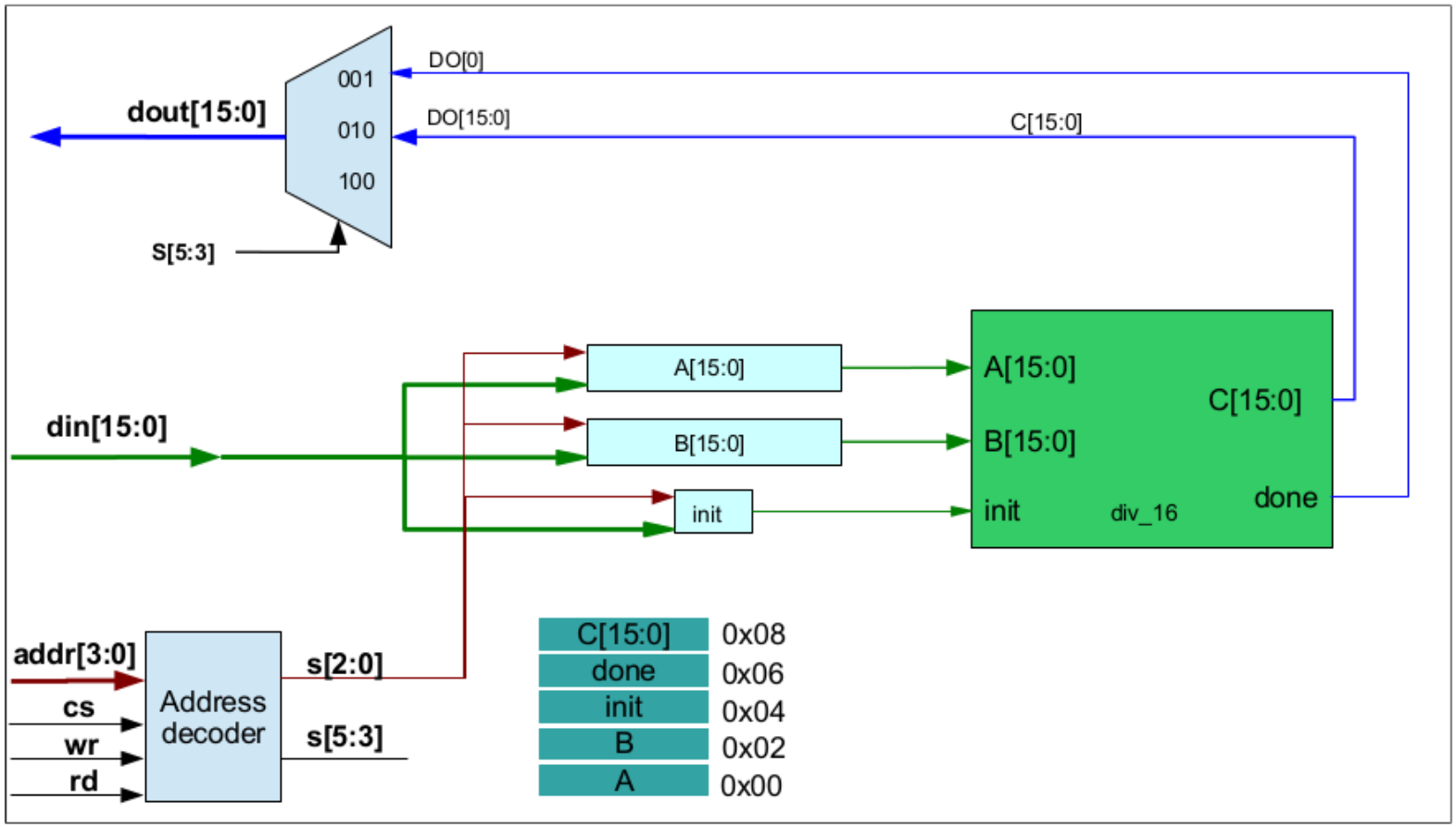

din[15:0]
A[15:0]
B[15:0]
init
A[15:0]
B[15:0]
init
PP[31:0]
done
mult_32
addr[3:0]
cs
wr
Address decoder
s0
s1
s2
always @(*)
if(cs & wr){
case (addr[3:0])
4'h0:begin s=5'b000001 ;end //A
4'h2:begin s=5'b000010 ;end //B
4'h4:begin s=5'b000100 ;end //init
default:begin s=5'b000000 ; end
endcase
}
init
B
A
0x04
0x02
0x00

Diagrama de Bloques Divisor



Interfaz basada en memoria