

Electrónica digital 1

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

Recordando

Hardware Software Interface

ejemplo el procesador J1

ejemplo Arquitectura SoC J1

Índice

Recordando

Hardware Software Interface

ejemplo el procesador J1

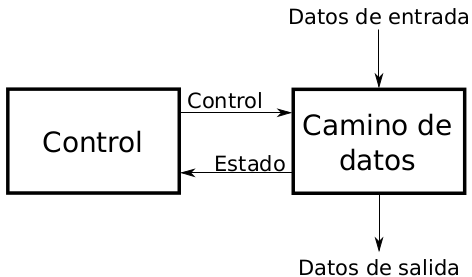
ejemplo Arquitectura SoC J1

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



Índice

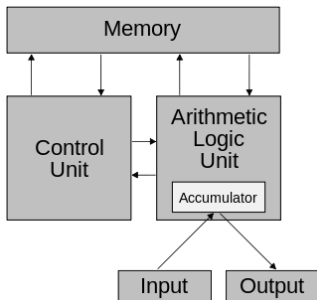
Recordando

Hardware Software Interface

ejemplo el procesador J1

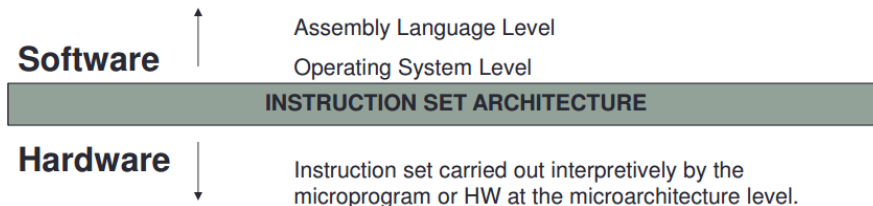
ejemplo Arquitectura SoC J1

Hardware Software Interface



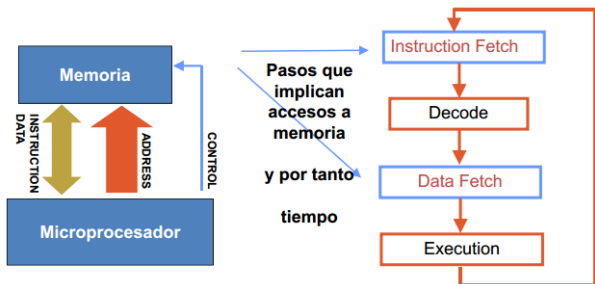
Hardware Software Interface

La frontera entre el hardware y el software



Cada instrucción es directamente ejecutada por el hardware

Ciclo de instrucción

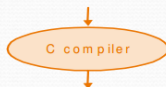


- ▶ Buscar la instrucción en la memoria principal
- ▶ Decodificar la instrucción
- ▶ Ejecutar la instrucción
- ▶ Almacenar o guardar resultados

Abstracción

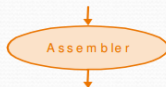
High-level
language
program
(in C)

```
swap(int v[], int k)
{int temp;
  temp = v[k];
  v[k] = v[k+1];
  v[k+1] = temp;
}
```



Assembly
language
program
(for MIPS)

```
swap:
  muli $2, $5, 4
  add $2, $4, $2
  lw $15, 0($2)
  lw $16, 4($2)
  sw $16, 0($2)
  sw $15, 4($2)
  jr $31
```



Binary machine
language
program
(for MIPS)

```
000000001010000100000000000011000
00000000100011100001100000100001
100011000110001000000000000000000
100011001111001000000000000000100
101011001111001000000000000000000
101011000110001000000000000000100
00000011111000000000000000001000
```

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

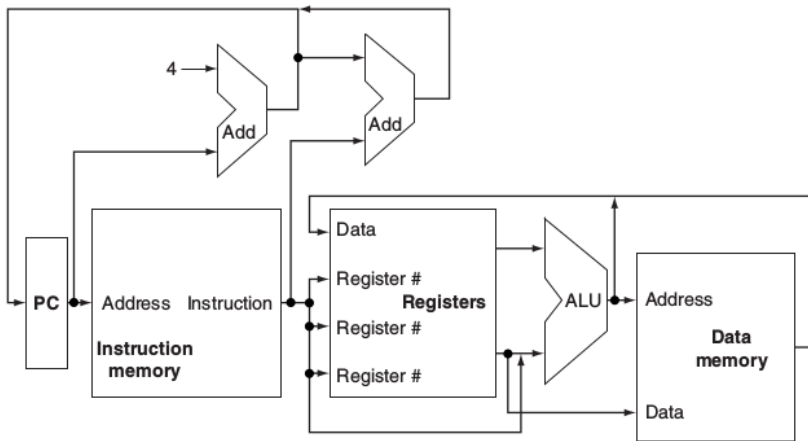
El procesador

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

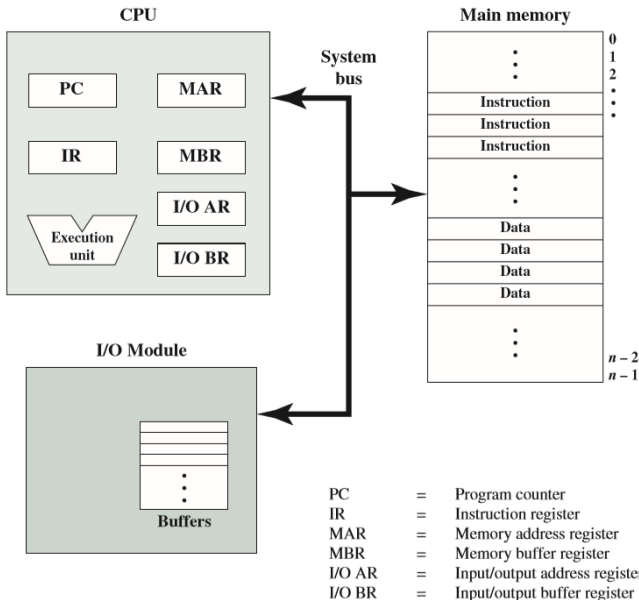
El procesador consta de dos componentes:

- ▶ 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



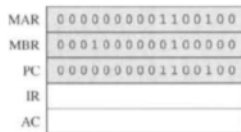
Ciclo de instrucción



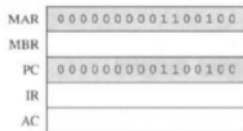
Ciclo de instrucción



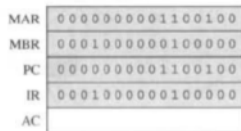
(a) Comienzo



(c) Segundo paso



(b) Primer paso



(d) Tercer paso

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

Unidad aritmético-lógica

La cual se encarga de realizar las operaciones que requiera el algoritmo.

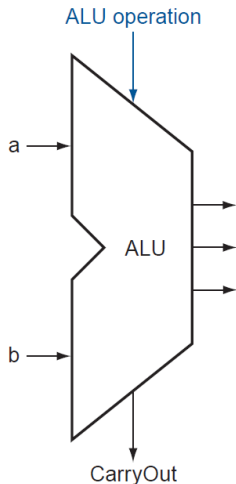
Lógicamente las arquitecturas no tienen implementadas todas las posibles funciones matemáticas o funciones aritméticas (instrucciones)

Banco de registros

Mantiene almacenada la información o los datos

Registros de acceso a memoria, registros programados, registro especiales etc.

Datapath (Componentes básicos)

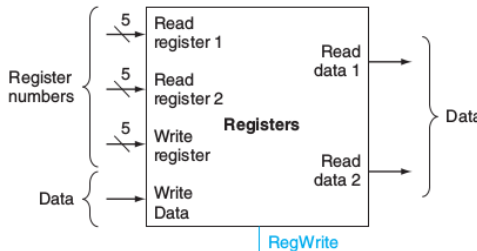


code	operation
0	T
1	N
2	$T + N$
3	$T \text{ and } N$
4	$T \text{ or } N$
5	$T \text{ xor } N$
6	$\sim T$
7	$N = T$
8	$N < T$
9	$N \text{ rshift } T$
10	$T - 1$
11	R
12	$[T]$
13	$N \text{ lshift } T$
14	depth
15	$Nu < T$

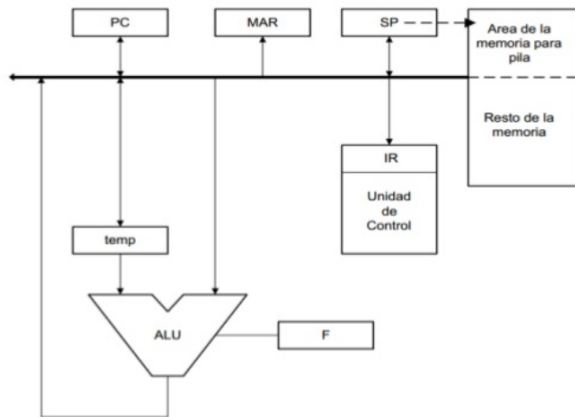
Banco de Registros

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)

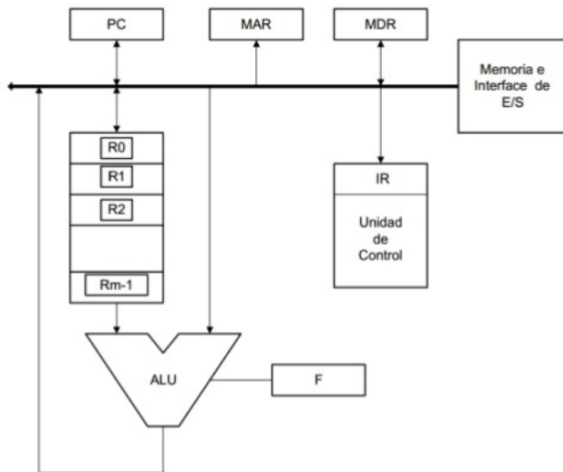


SP: Puntero de la Pila: registro con la dirección de la última palabra insertada en la pila **TOS, Top of Stack**
en función de Sp se obtiene el próximo registro **NOS Next of Stack**

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

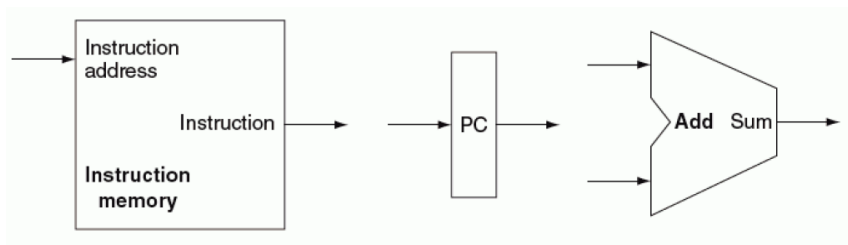


Banco de registros de propósito general ($R_0 \dots R_{m-1}$)
Maquinas de dos (Lectura destructiva) o tres operandos

Explícito

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

Componentes básicos adicionales

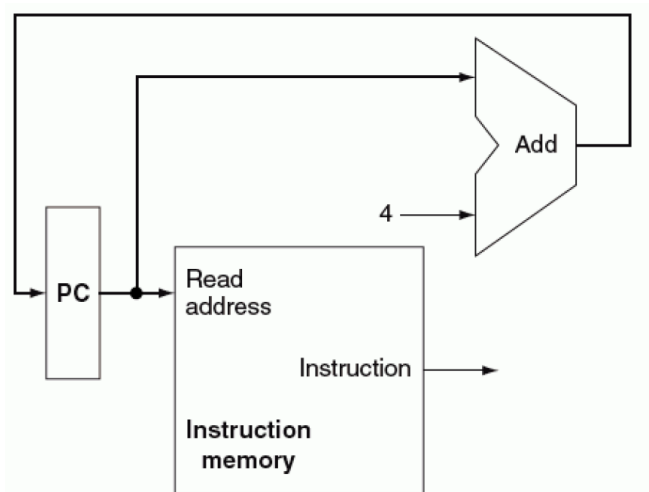


Una memoria para guardar y leer instrucciones

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

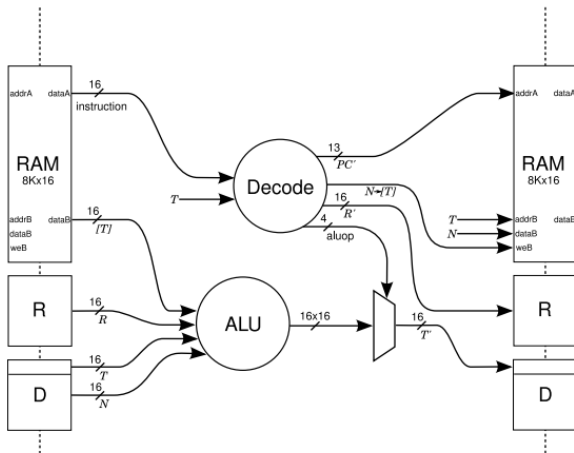
Recordando

Hardware Software Interface

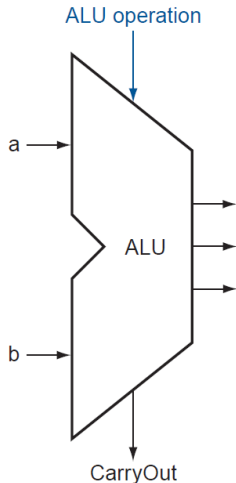
ejemplo el procesador J1

ejemplo Arquitectura SoC J1

Datapath J1 (Componentes básicos)

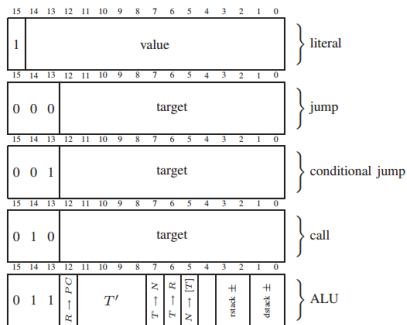


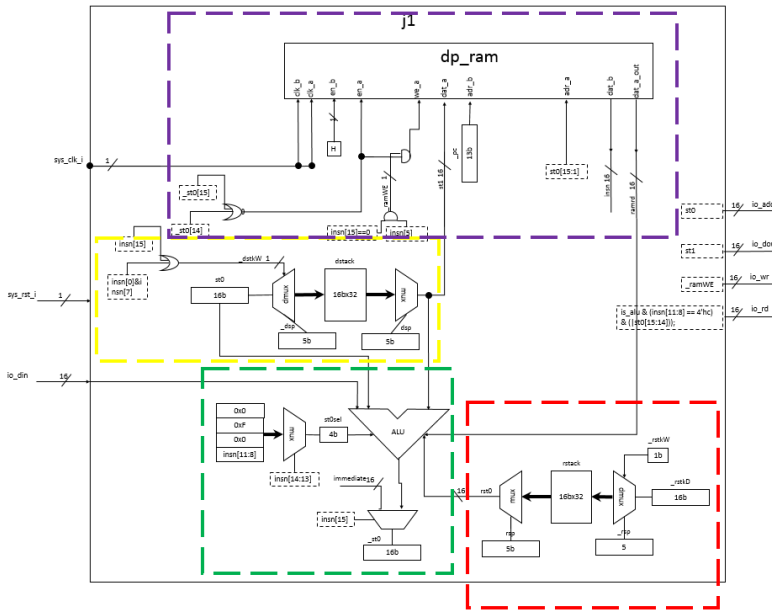
Datapath (Componentes básicos)



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

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





Índice

Recordando

Hardware Software Interface

ejemplo el procesador J1

ejemplo Arquitectura SoC J1

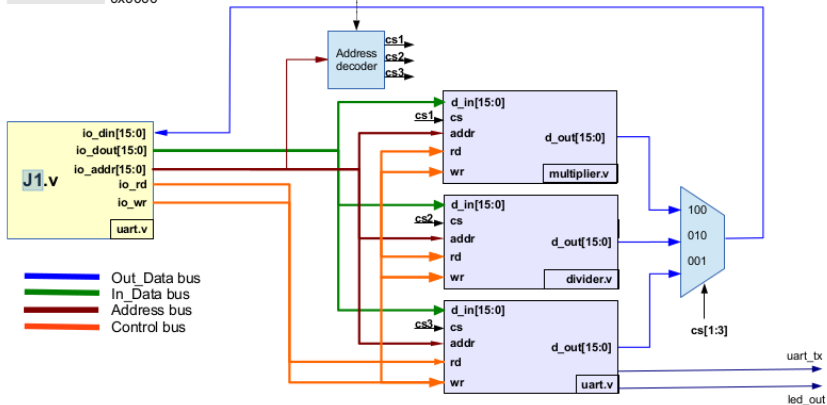
J1 CPU

	0xFF00
UART	0x6900
Divider	0x6800
Multiplier	0x6700
	0x0000

```

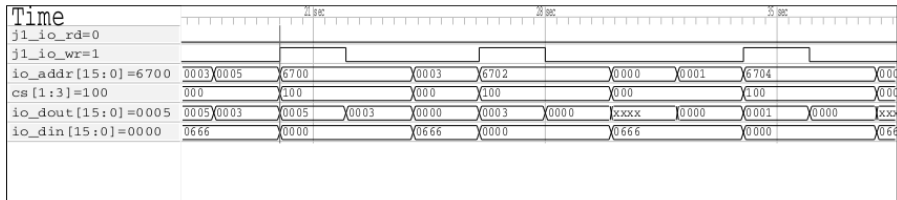
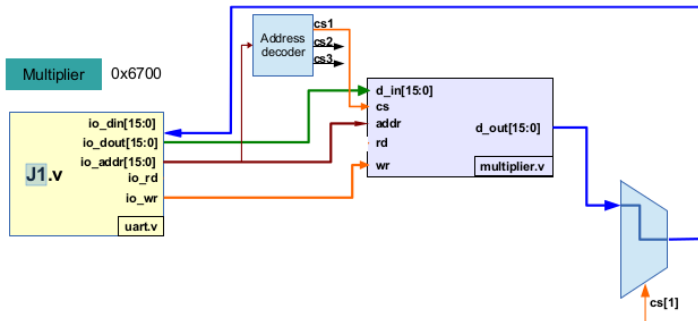
always @*
begin
  case (j1_io_addr[15:8]) // direcciones - chip_select
    8'h67: cs= 3'b100;      //mult
    8'h68: cs= 3'b010;      //div
    8'h69: cs= 3'b001;      //uart
    default: cs= 3'b000;
  endcase
end

```



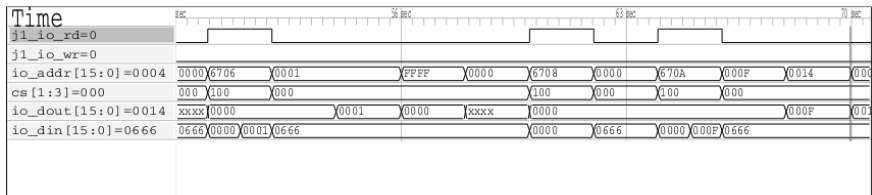
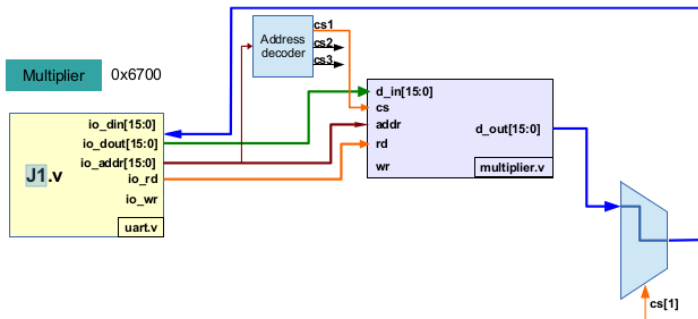
J1 CPU escritura

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



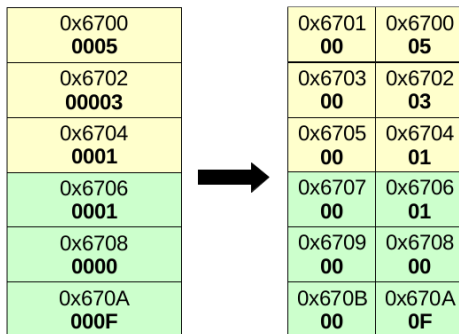
J1 CPU leer

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



J1 CPU leer

Almacenamiento por bytes



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

Mapa de memoria Multiplicador

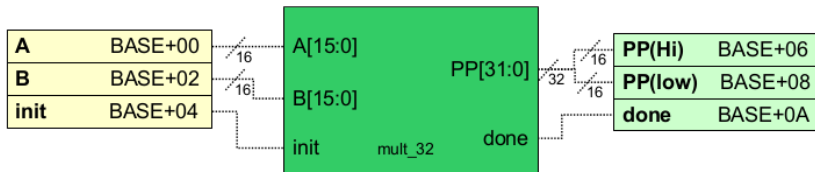
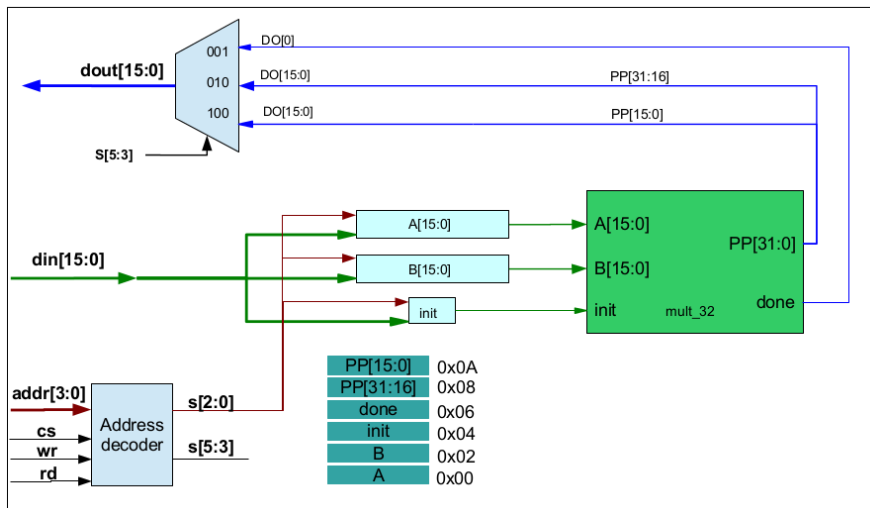
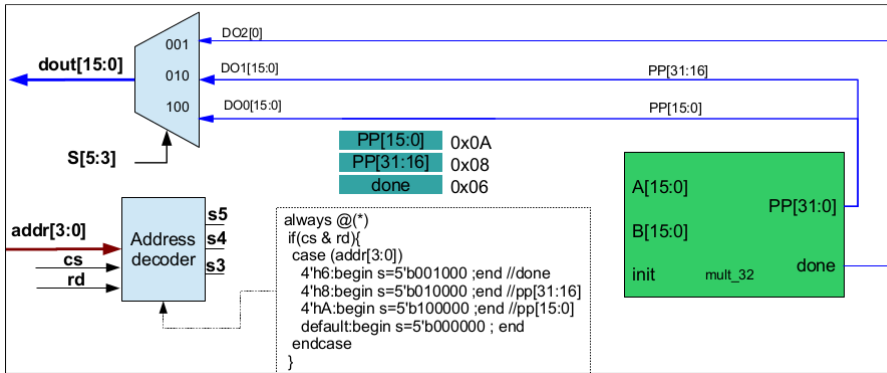


Diagrama de bloques Multiplicador



Lectura



Escritura

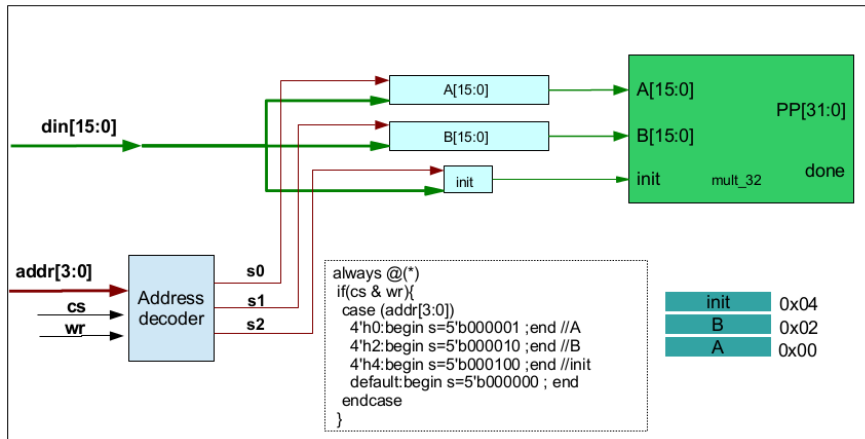
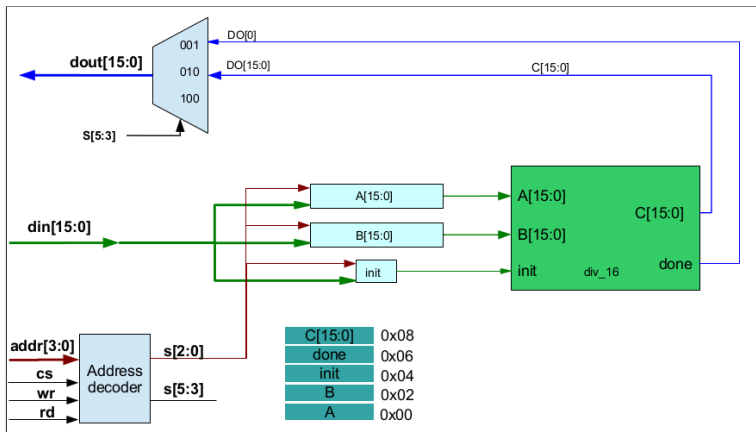


Diagrama de Bloques Divisor



Interfaz basada en memoria

