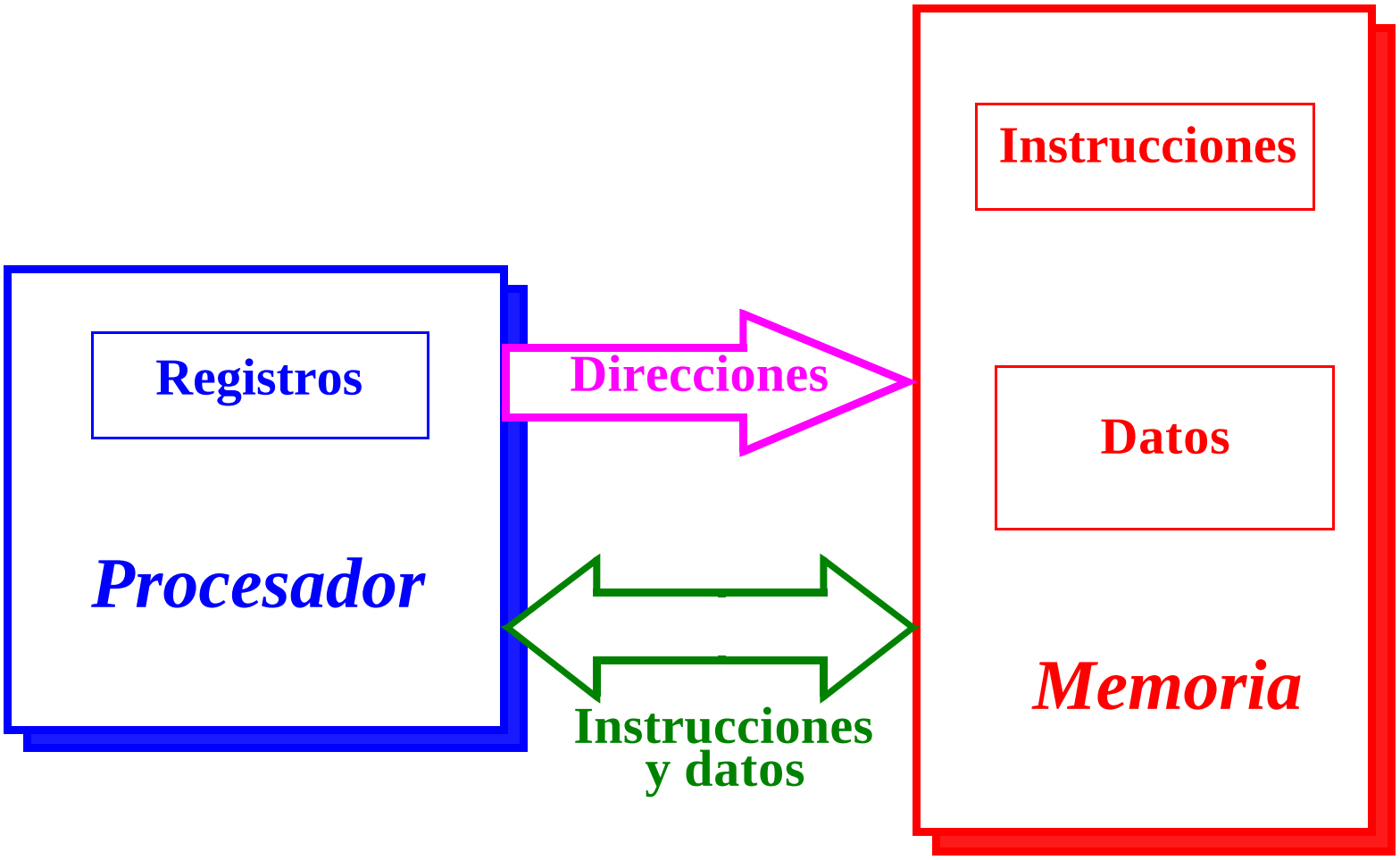
Ejemplo de un Procesador: MU0



```
float fir_filter(float input, float *coef, int n, float *history
{
        int i;
        float *hist_ptr, *hist1_ptr, *coef_ptr;
        float output;
        hist_ptr = history;
        hist1_ptr = hist_ptr;           /* use for history update */
        coef_ptr = coef + n - 1;        /* point to last coef */
/*form output accumulation */
        output = *hist_ptr++ * (*coef_ptr_);
        for(i = 2; i < n; i++)
        {
                *hist1_ptr++ = *hist_ptr; /* update history array */
                output += (*hist_ptr++) * (*coef_ptr-);
        }
        output += input * (*coef_ptr);  /* input tap */
        *hist1_ptr = input;             /* last history */
        return(output);
}
```

Elementos del MU0

- **Componentes:**
  Un conjunto de dispositivos que configuran la estructura del MU0
- **Conjunto de Instrucciones:**
  Las operaciones que pueden ejecutarse con la estructura del MU0
- **Configuración Lógica:**
  La estructura operativa del MU0, formada por dos elementos:
  - El Camino de Datos
  - La Lógica de Control

❑ **MU0 tiene 16 bits, con 12 bits de espacio de direcciones:**

Componentes del MU0

- Contador de Programa (PC)
- Acumulador (ACC)
- Unidad Aritmético-Lógica (ALU)
- Registro de Instrucciones (IR)
- Lógica de control y de decodificación de instrucciones

Instrucciones del MU0

Instrucción	Cod.Oper.	Efecto
LDA S	0000	ACC:=mem_{16}[S]
STO S	0001	mem_{16}[S]:=ACC
ADD S	0010	ACC:=ACC+mem_{16}[S]
SUB S	0011	ACC:=ACC-mem_{16}[S]
JMP S	0100	PC:=S
JGE S	0101	if ACC $\geq$ 0 PC:=S
JNE S	0110	if ACC $\not=$ 0 PC:=S
STP	0111	stop

Tipos de Instrucciones

- Procesamiento de datos (ej. ADD, SUB)
- Movimiento de datos (copian datos de un lugar de la memoria a otro o de la memoria a los registros del procesador, ej. STO, LDA)
- Control de flujo (cambia la ejecución de una parte del programa a otra diferente, ej. JMP)
- Especiales (controlan el estado de ejecución del procesador, ej. STP)
- Puede haber instrucciones que entren en más de una categoría

Direccionamiento de Instrucciones

Toda instrucción precisa:

- el nombre de la instrucción
- el lugar de memoria donde están los operandos
- el lugar de memoria donde debe guardarse el resultado
- la dirección de la próxima instrucción a ejecutar

Forma "natural" de una instrucción: 4-direcciones

Ejemplo: ADD d, s1, s2, next_i; d:=s1+s2

Direccionamiento simplificado de Instrucciones

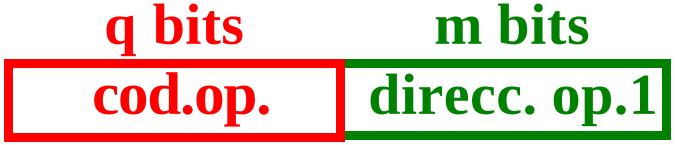
Instrucción 3-direcciones: suponiendo que la próxima instr. está en la dirección siguiente

Ejemplo: ADD d, s1, s2, ; d:=s1+s2

Instrucción 2-direcciones: suponiendo que el registro de destino es el mismo que el registro fuente

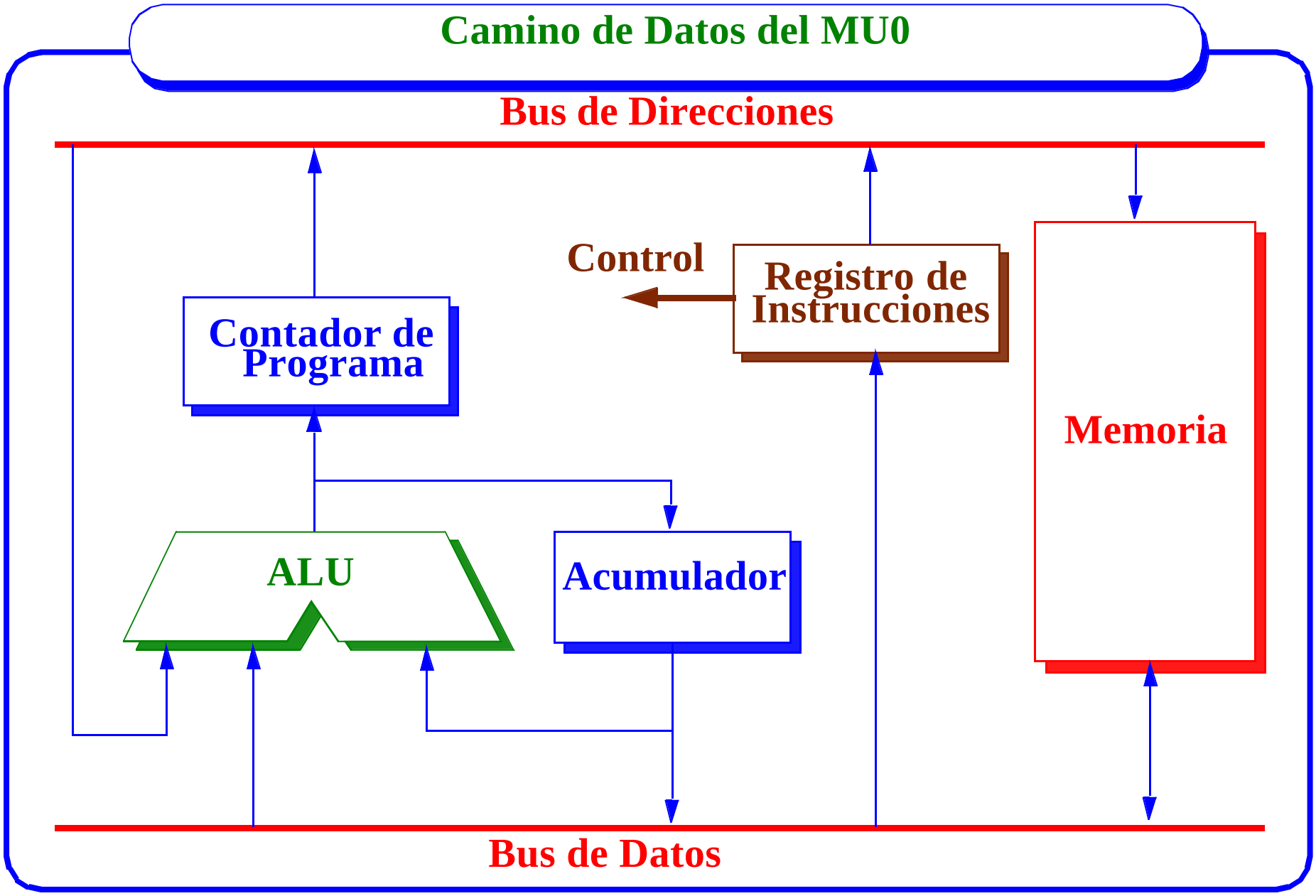
Ejemplo: ADD d, s1 ; d:=d+s1

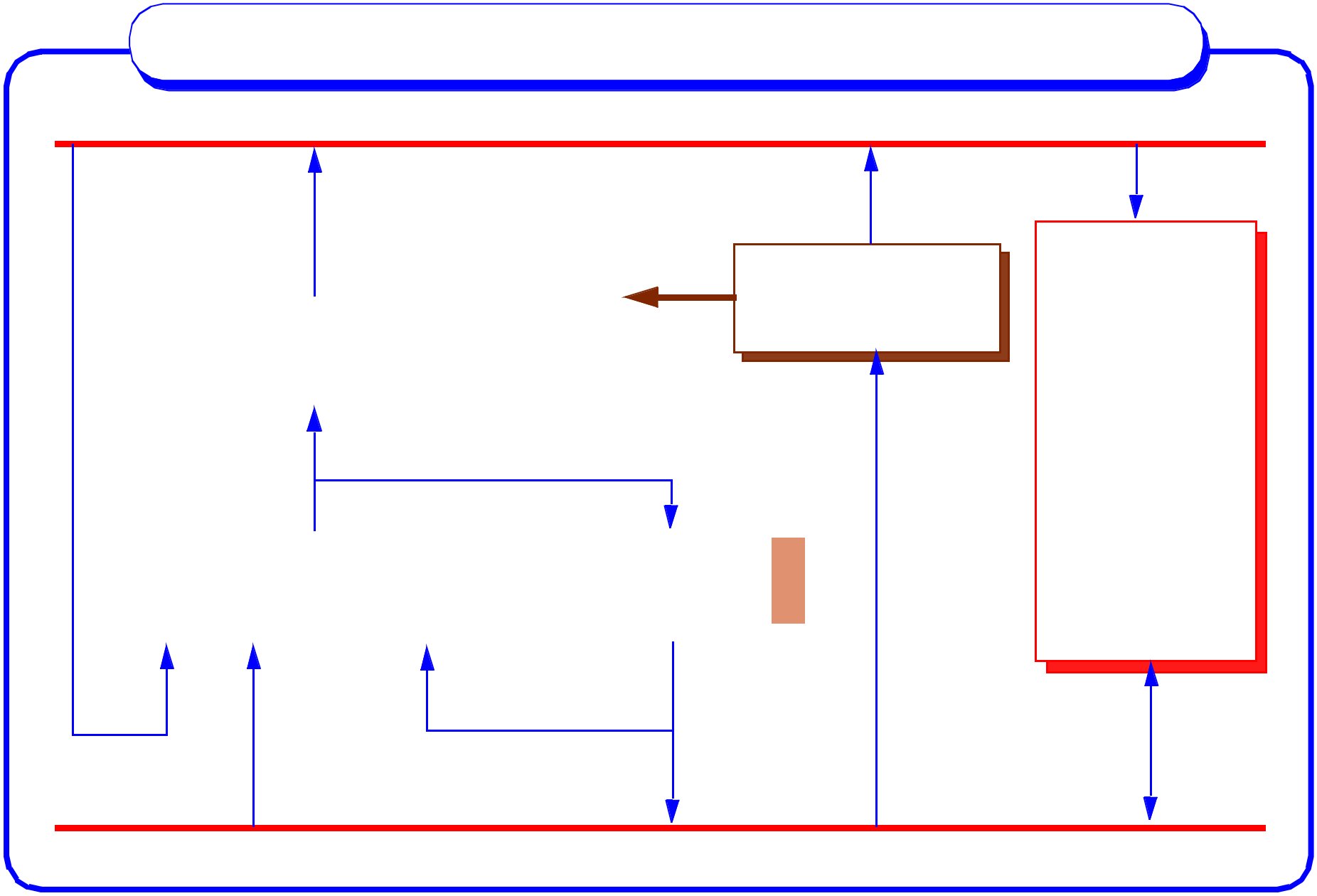
Instrucción 1-dirección: suponiendo implícito el registro de destino



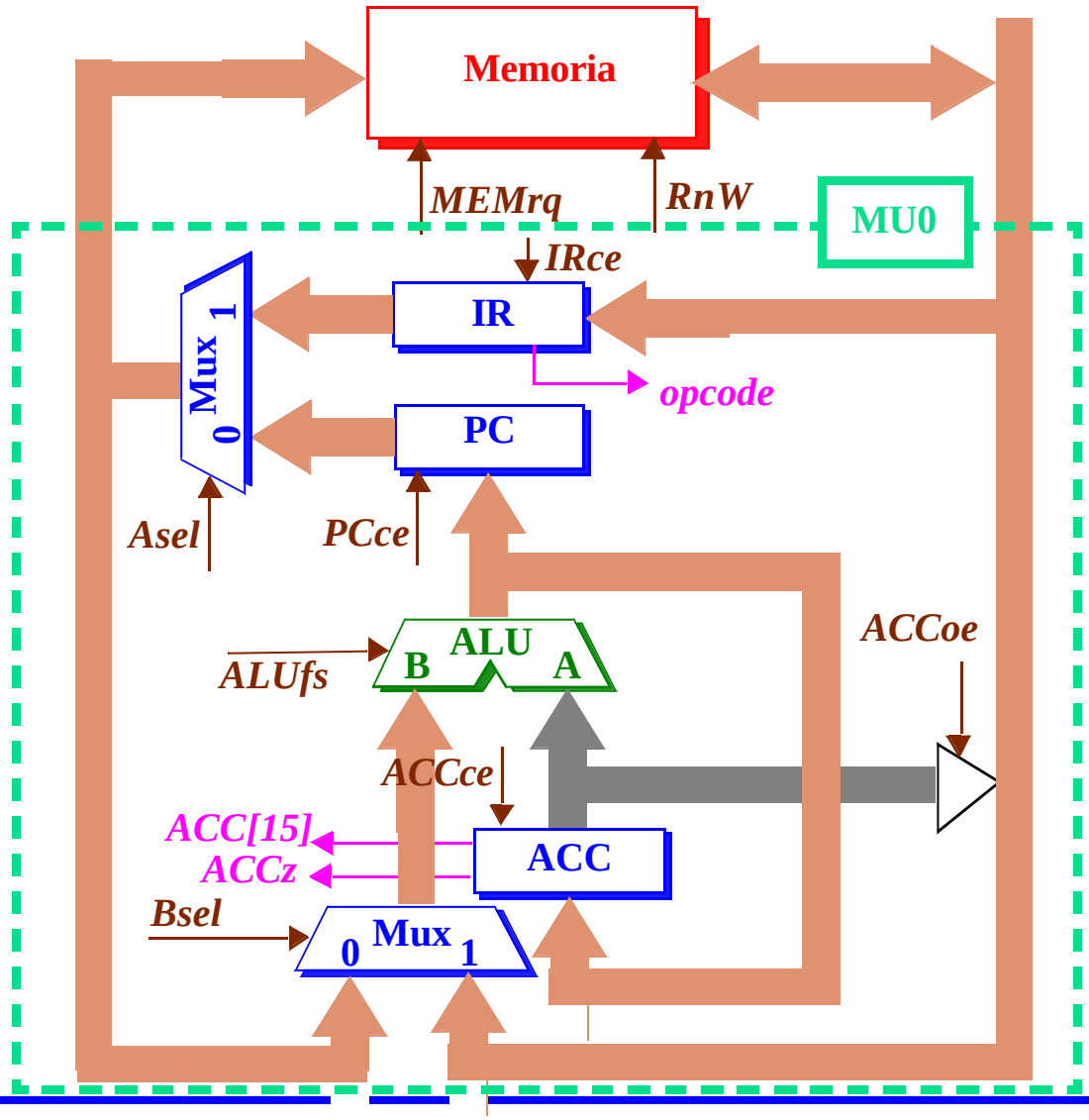
Ejemplo: ADD s1 ; acumulador:=acumulador+s1

Camino de Datos del MU0





Organización a nivel de Transferencias de Registro del MU0



Reset

Ex/ft (execute/fetch)

Unidad de Control del MU0: Comandos de cada Instrucción

		a	b	c	d	a	β	γ	δ	ε	ϕ	η	φ	κ	λ
Reset	xxxx	1	x	x	x	0	0	1	1	1	0	= 0	1	1	0
LDA S	0000	0	0	x	x	1	1	1	0	0	0	= B	1	1	1
	0000	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
STO S	0001	0	0	x	x	1	x	0	0	0	1	x	1	0	1
	0001	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
ADD S	0010	0	0	x	x	1	1	1	0	0	0	A+B	1	1	1
	0010	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
SUB S	0011	0	0	x	x	1	1	1	0	0	0	A - B	1	1	1
	0011	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
JMP S	0100	0	x	x	x	1	0	0	1	1	0	B+1	1	1	0
JGE S	0101	0	x	x	0	1	0	0	1	1	0	B+1	1	1	0
	0101	0	x	x	1	0	0	0	1	1	0	B+1	1	1	0
JNE S	0110	0	x	0	x	1	0	0	1	1	0	B+1	1	1	0
	0110	0	x	1	x	0	0	0	1	1	0	B+1	1	1	0
STP	0111	0	x	x	x	1	x	0	0	0	0	x	0	1	0

Entradas **Salidas**

Entradas

a	Reset
b	Ex/ft
c	ACCz
d	ACC15

Salidas

α	Asel
α	Bsel
β	ACCce
γ	PCce
δ	IRce
ε	ACCoe
ϕ	ALUfs
η	MEMrq
φ	RnW
κ	Ex/ft

Evolución

Hasta 1980-85: Sólo sistemas CISC

- Instruciones muy complejas
- Uso de microcódigo en ROM para implementar las operaciones más frecuentes.
- Instrucciones de tamaño variable y nuchos formatos diferentes
- Valores en memoria pueden ser utilizados directamente como operandos
- Cada instrucción requiere muchos ciclos de reloj

Desde 1985: Aparecen sistemas RISC

- Instruciones más simples
- Instruciones de tamaño fijo (32-b) y pocos formatos distintos
- Un banco de registros (32 de 32-b)
- Las instrucciones que procesan datos operan sólo sobre los registros
- Estas instrucciones están “separadas” de las que acceden a memoria
- Cada instrucción requiere un único ciclo de reloj
- La lógica de decodificación se hace con puertas lógicas
- Las instrucciones se ejecutan en “pipeline”

Ejecución de Instrucciones

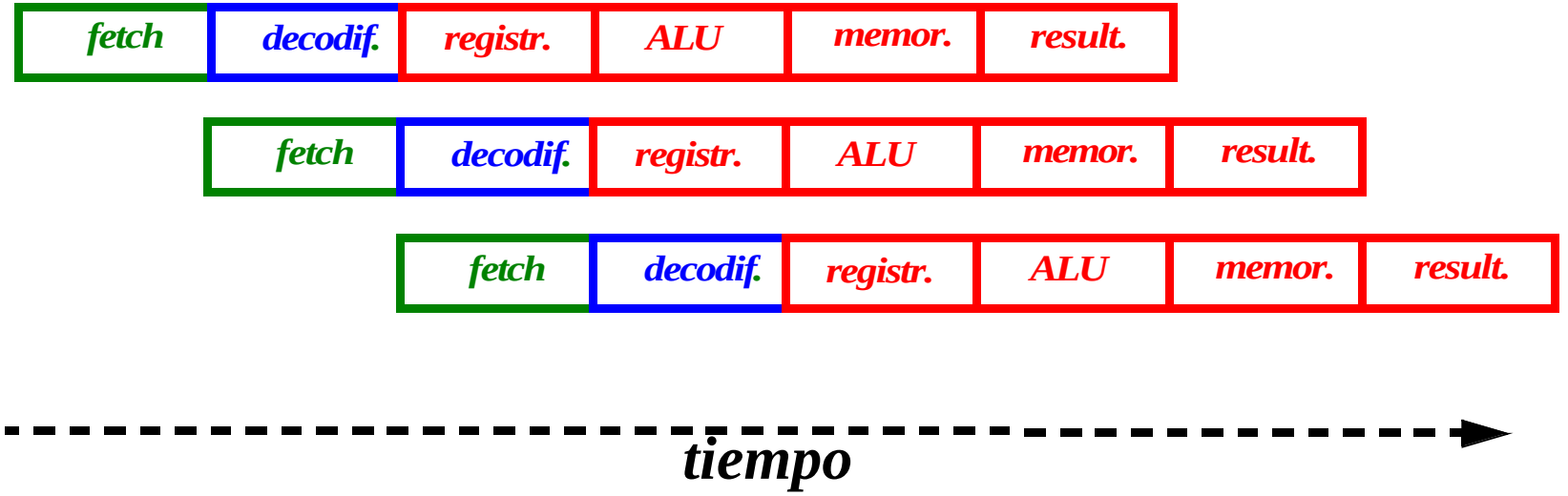
Ejecución de una instrucción típica: 6 pasos

1.- Sacar la instrucción de la memoria
2.- Decodificar para saber de qué instrucción se trata
3.- Acceder a los operandos que puedan ser necesarios desde el banco de registros
4.- Combinar los operandos para formar:
    a) el resultado
    b) una dirección de memoria
5.- Acceder a la memoria para obtener un dato-operando
6.- Escribir el resultado en el banco de registros

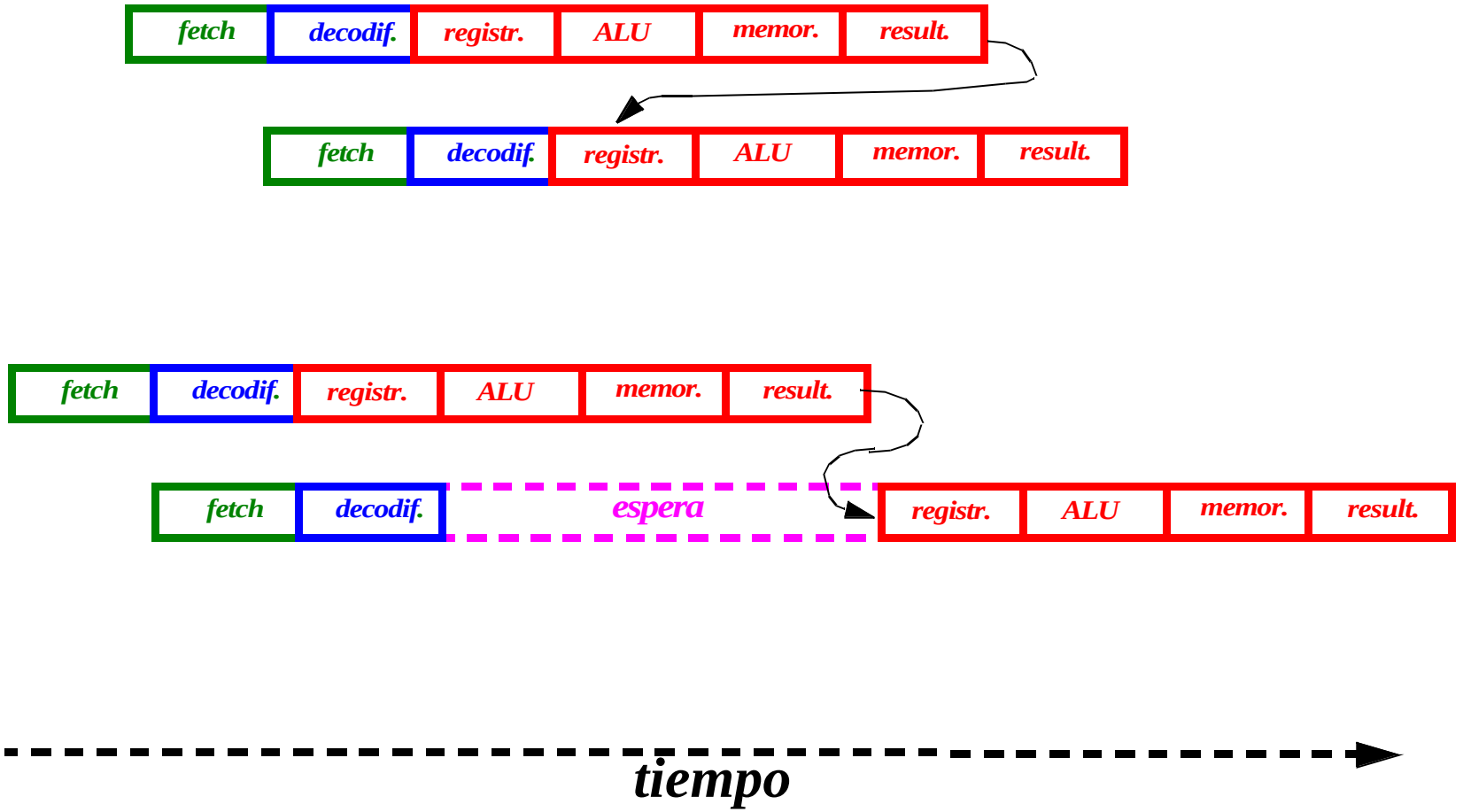
Pipelining

Métodos para acelerar la ejecución de instrucciones:

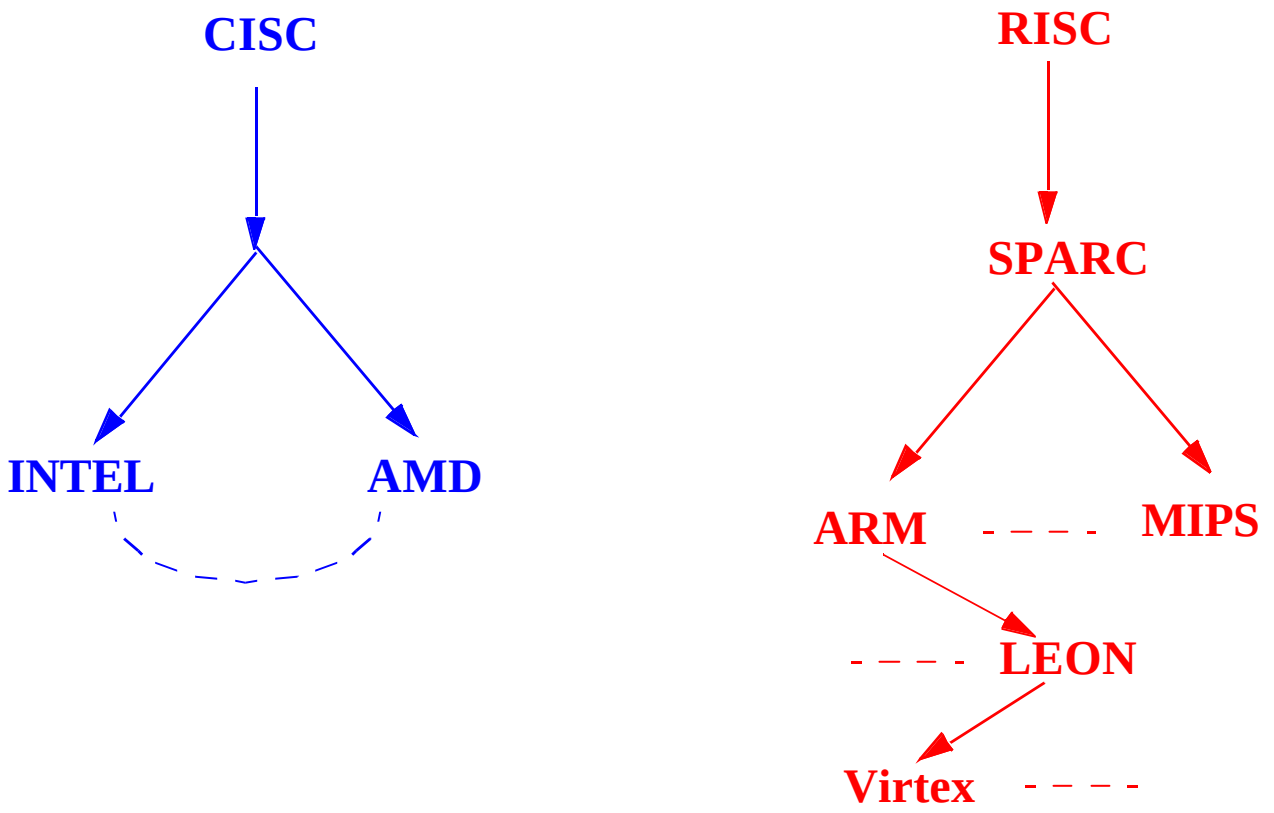
a.- Varios/muchos registros en “pipeline”
b.- Caché
3.- Instrucciones super-escalares

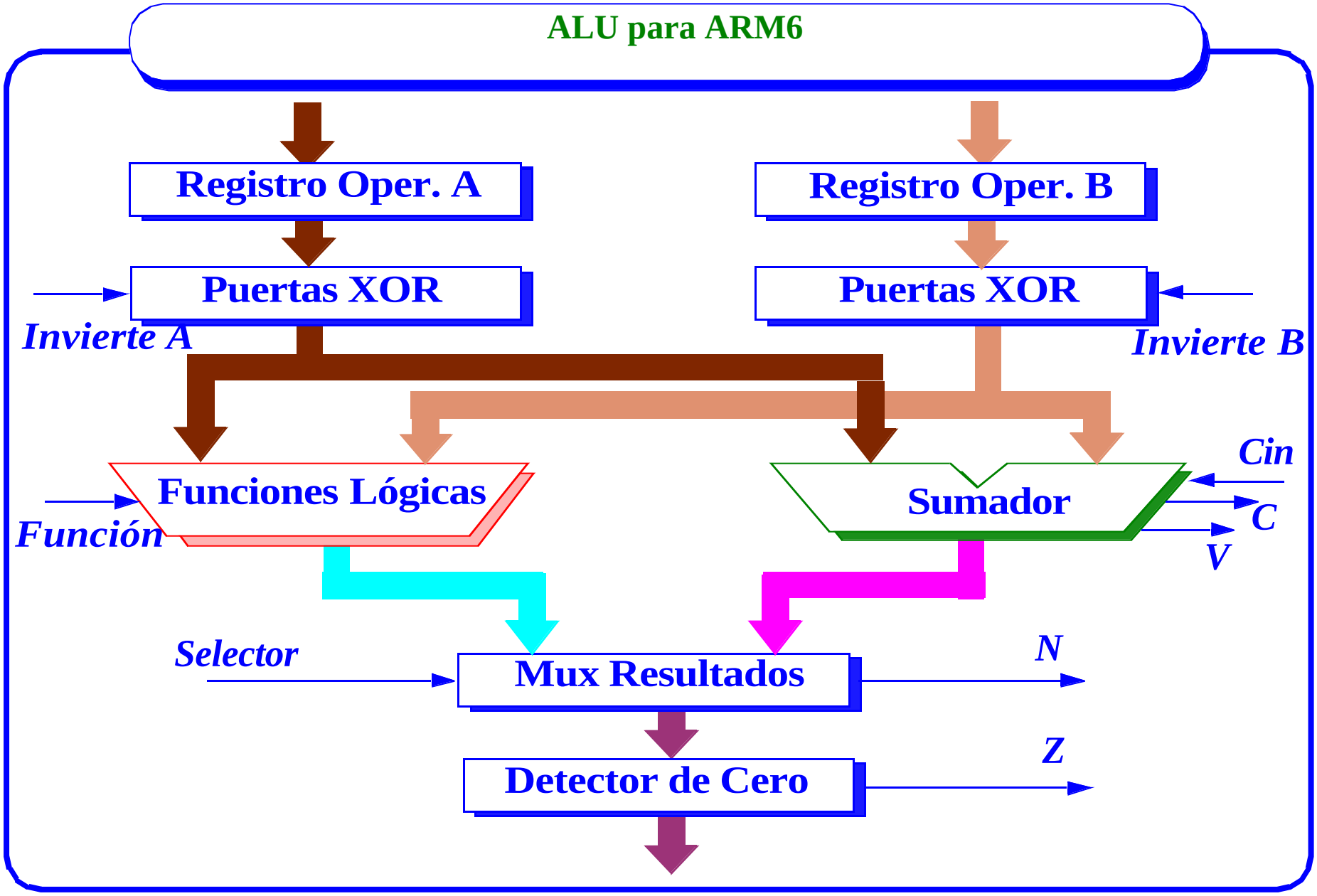


Pipelining: Conflictos

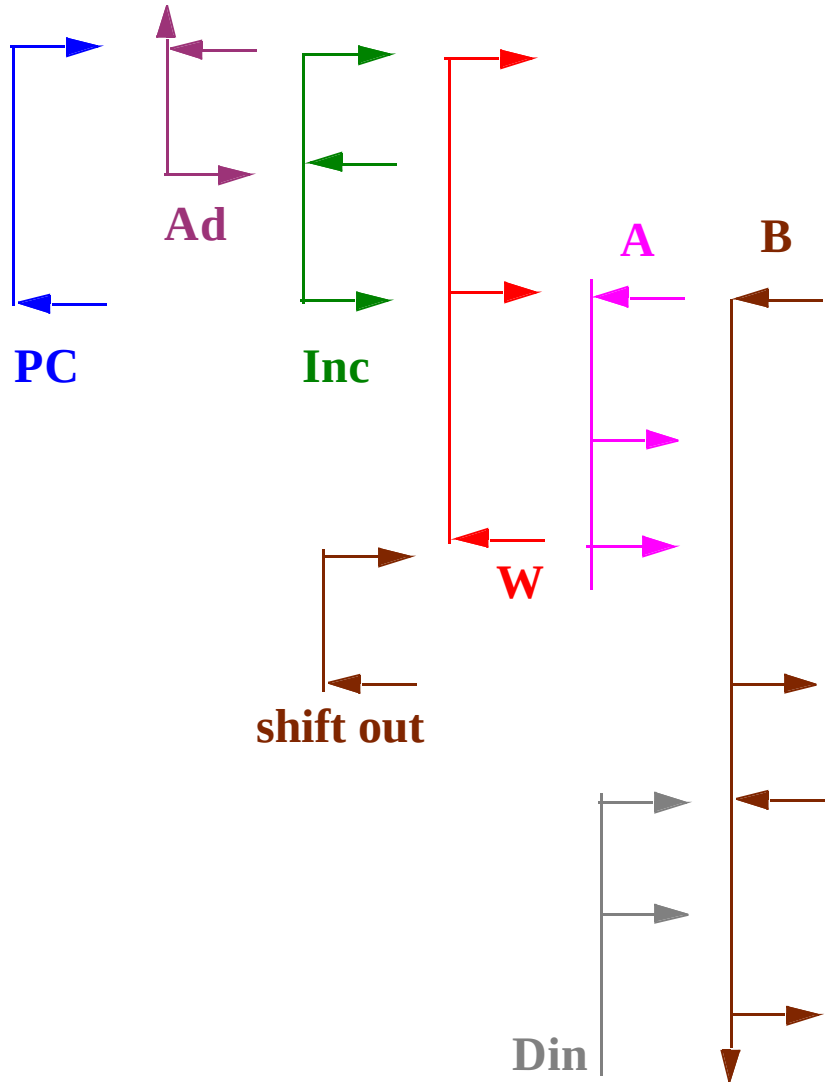


Evolución de los Sistemas de Tratamiento de Información



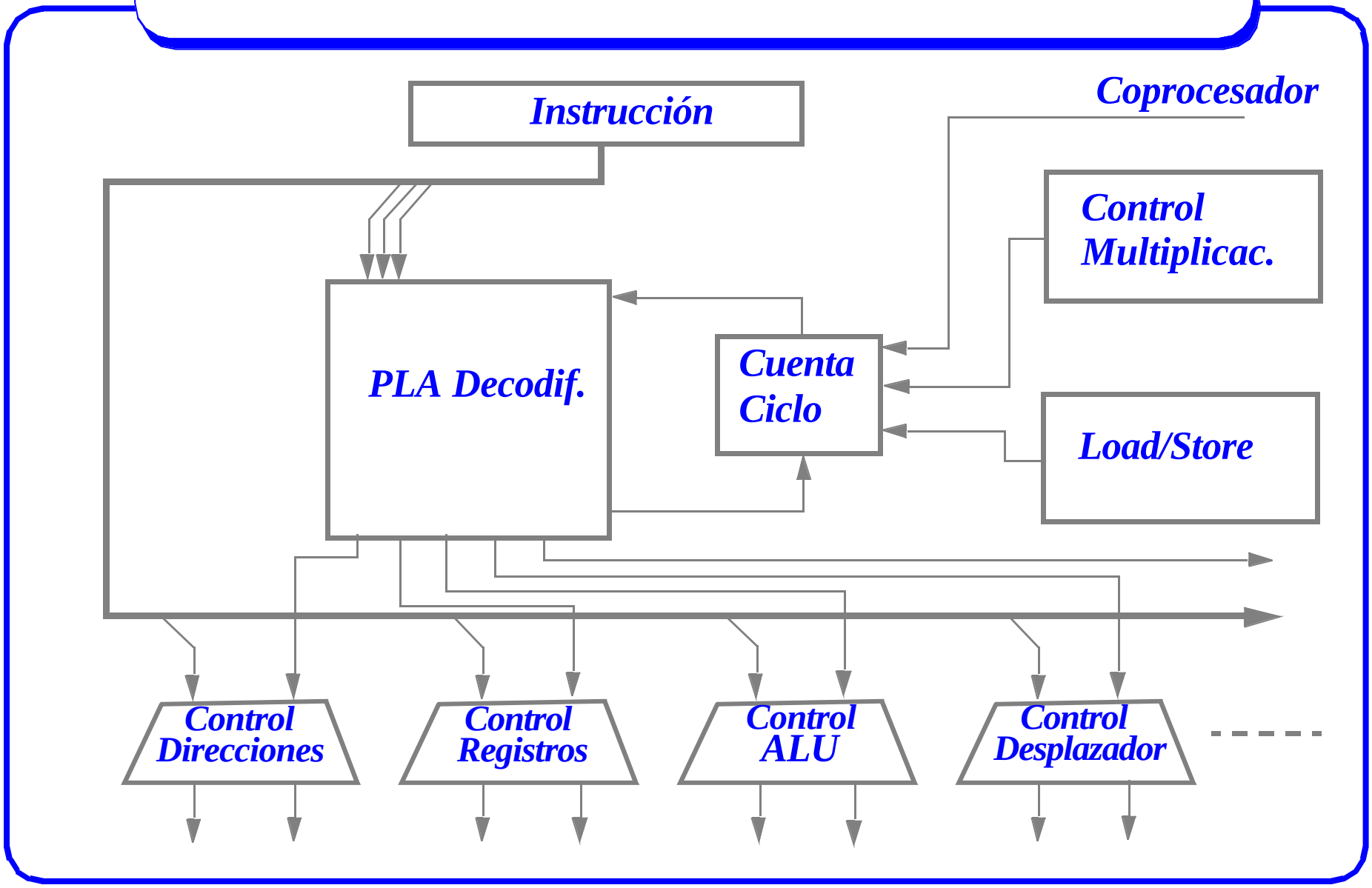

ALU para ARM6
Registro Oper. A
Registro Oper. B
Puertas XOR
Puertas XOR
Invierte A
Invierte B
Funciones Lógicas
Sumador
Cin
C
V
Función
Selector
Mux Resultados
N
Detector de Cero
Z

Buses y Registros en ARM6

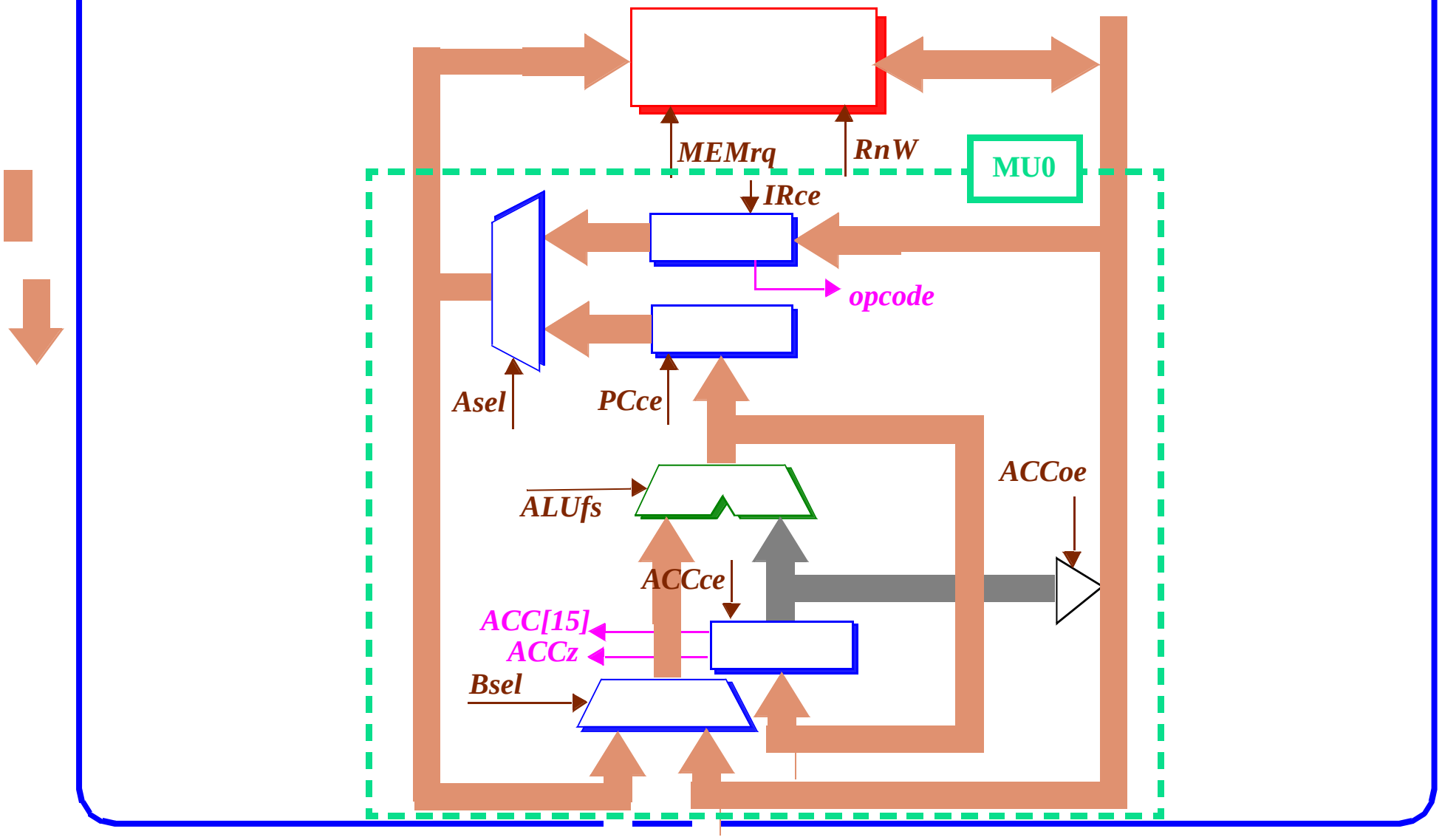


- Registro de Direcciones
- Incrementador
- Banco de Registros
- Multiplicador
- ALU
- Desplazador
- Data In
- Tubería de Instrucciones
- Data Out

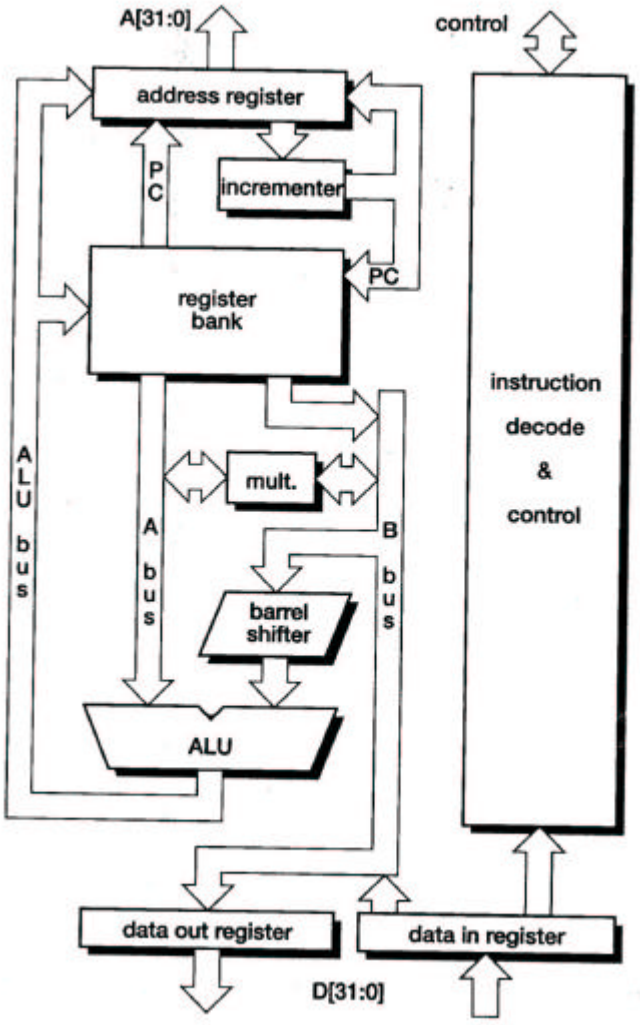
Estructura de Control de ARM6



Organización a nivel de Transferencias de Registro del MU0

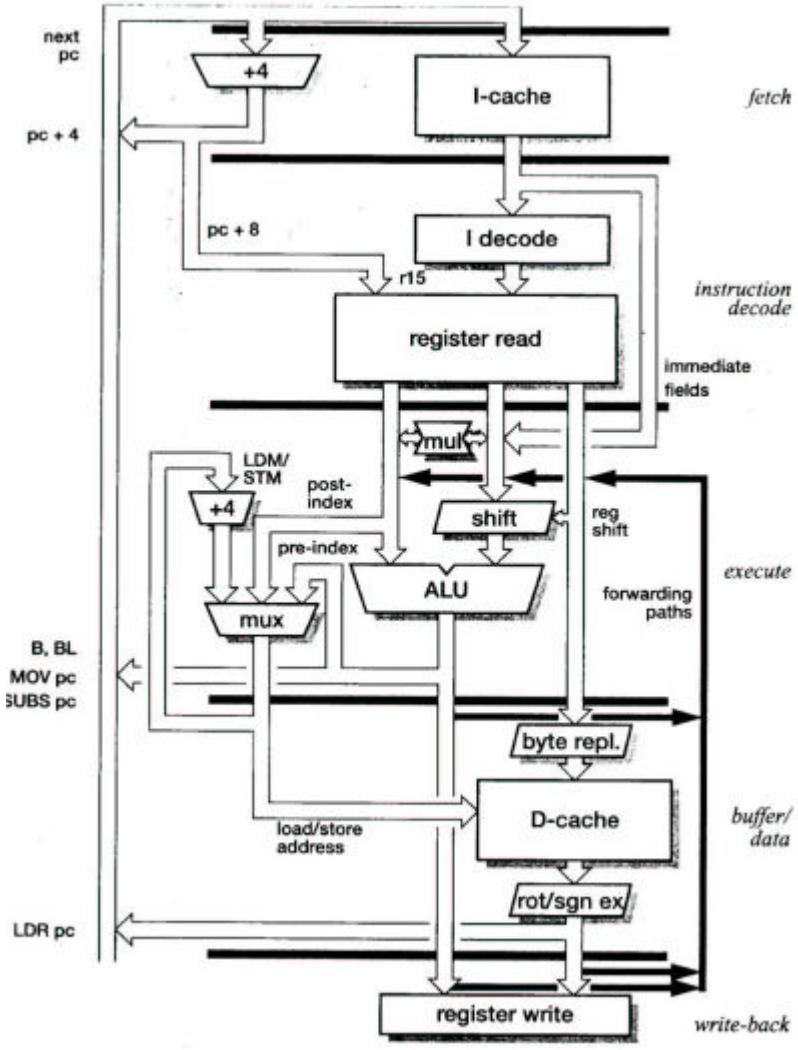


Organización de ARM con pipeline de 3 etapas

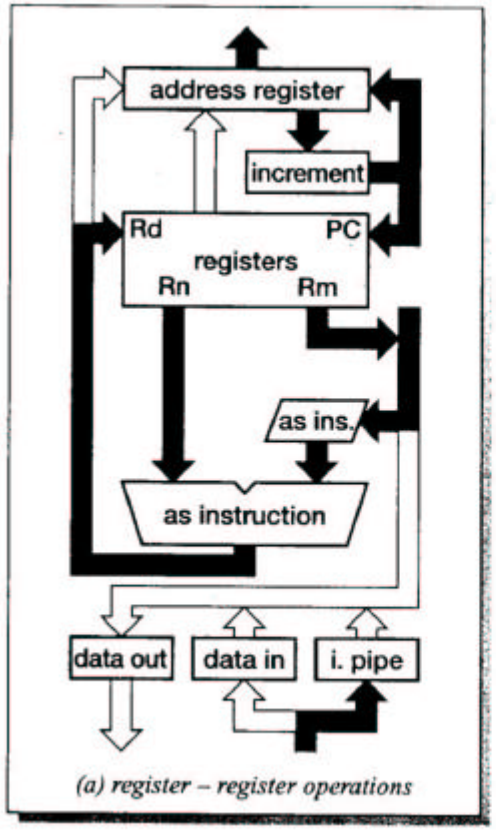


- Un banco de registros
- Un desplazador/rotador (barrel shifter)
- Una ALU
- Registro de direcciones e increm.
- Registros de datos
- Decodificador de instrucciones
- Lógica de control

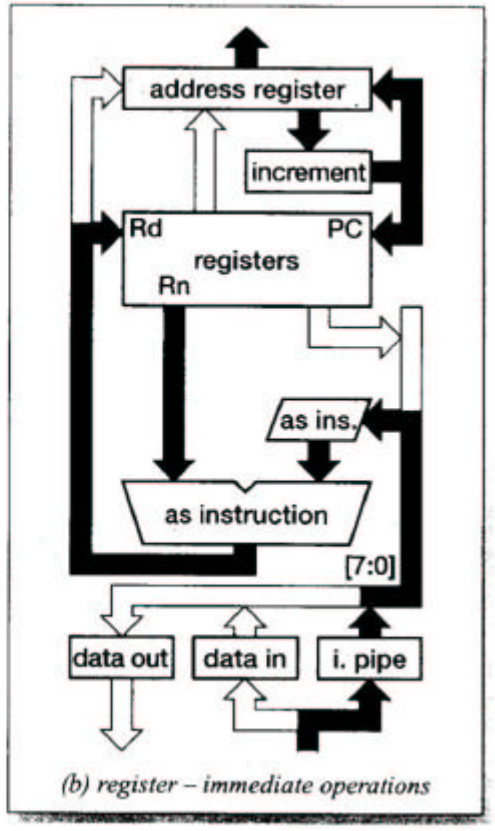
Organización de ARM con pipeline de 5 etapas



Actividad de una instrucción de procesamiento de datos

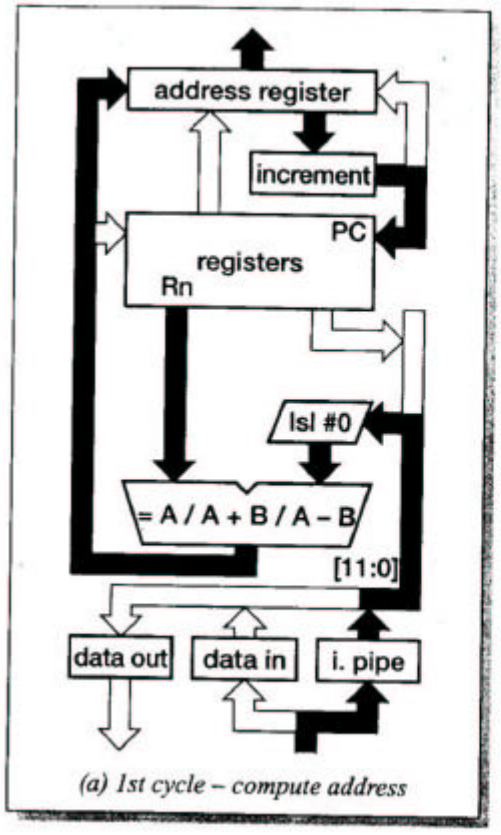


(a) register – register operations

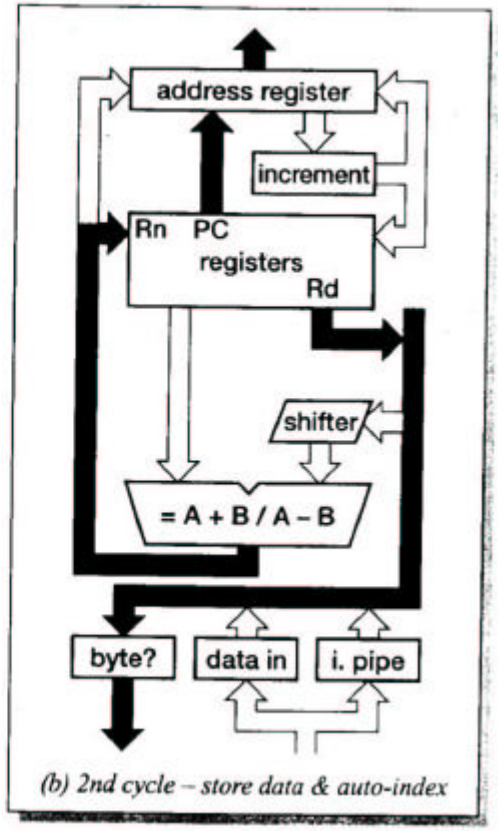


(b) register – immediate operations

Actividad de una instrucción de almacenamiento

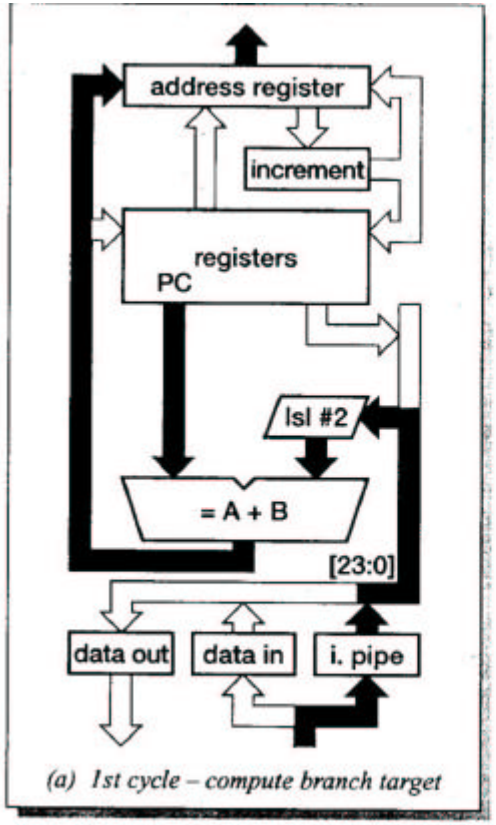


(a) 1st cycle – compute address

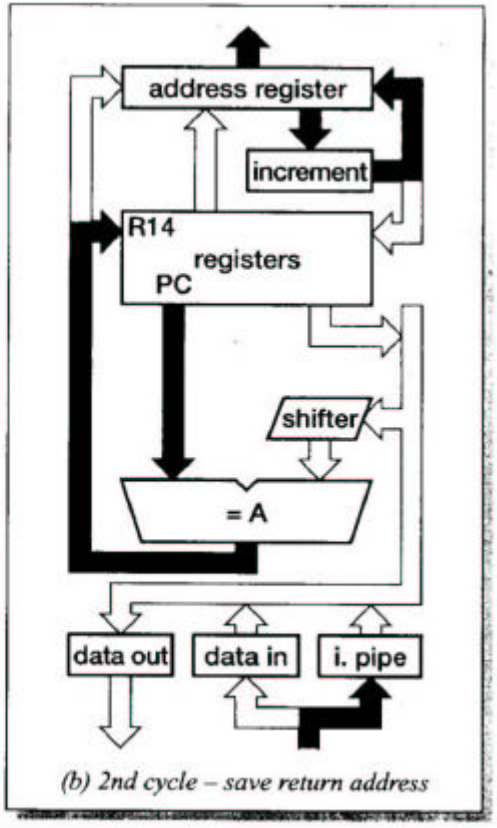


(b) 2nd cycle – store data & auto-index

Los dos primeros ciclos de actividad de una instrucción de salto

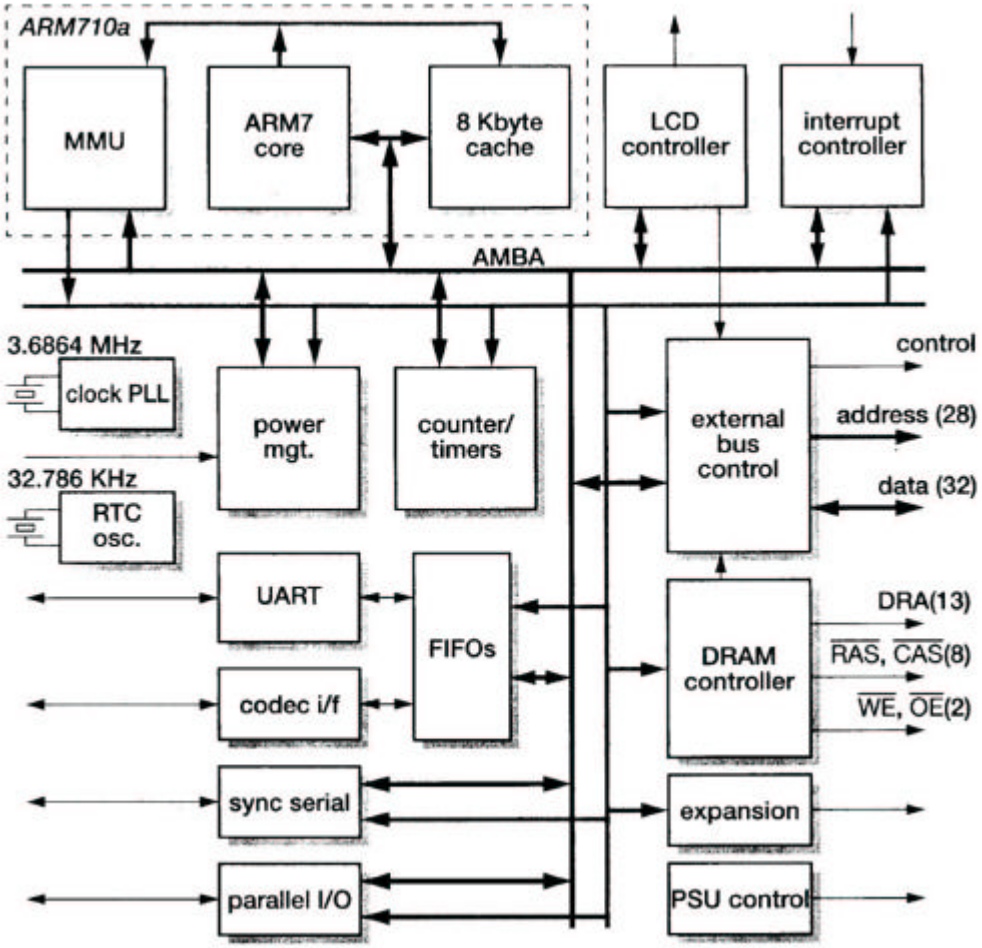


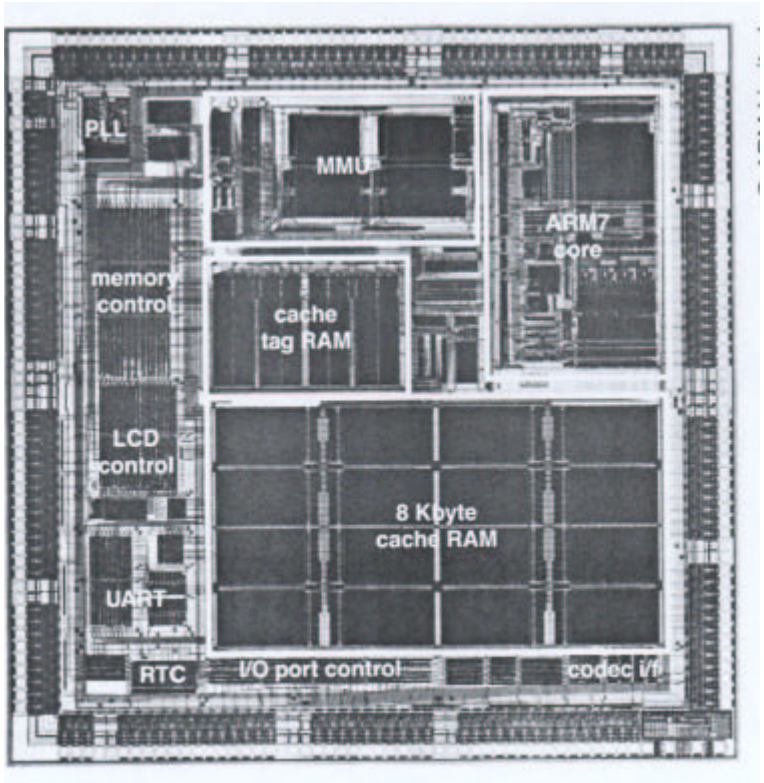
(a) 1st cycle – compute branch target



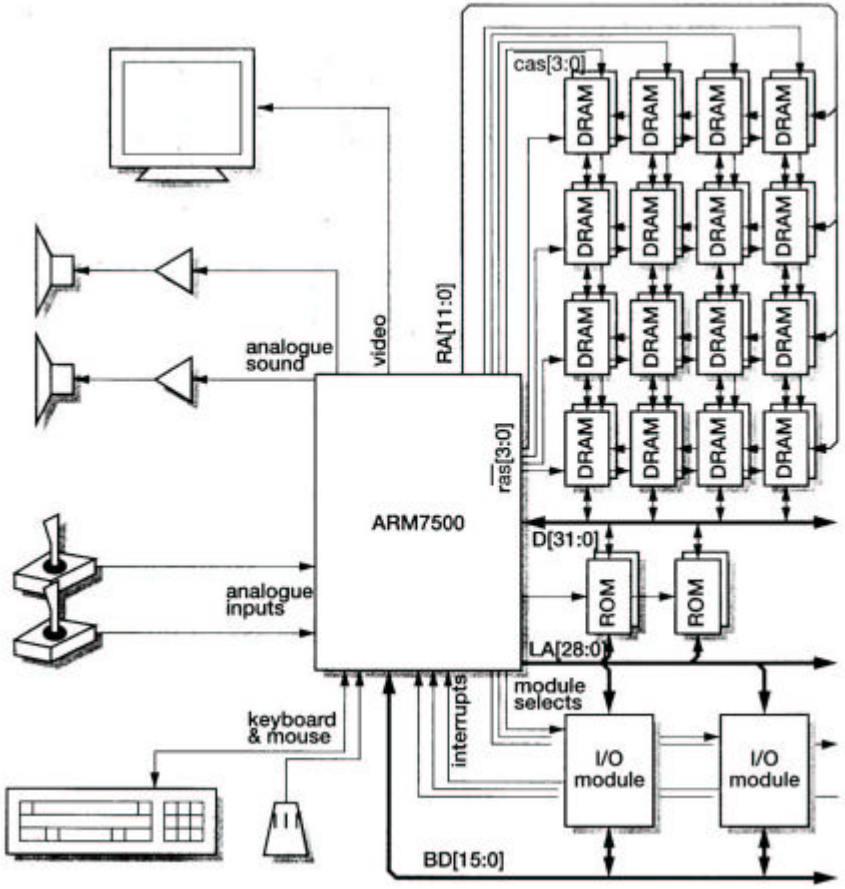
(b) 2nd cycle – save return address

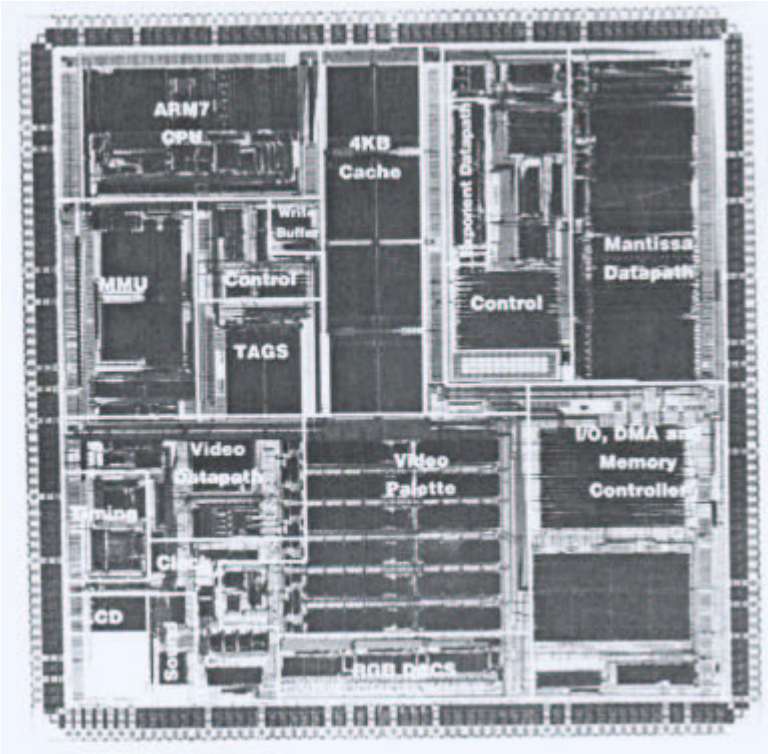
Ejemplo: ARM7100





Ejemplo: ARM7500 en un sistema

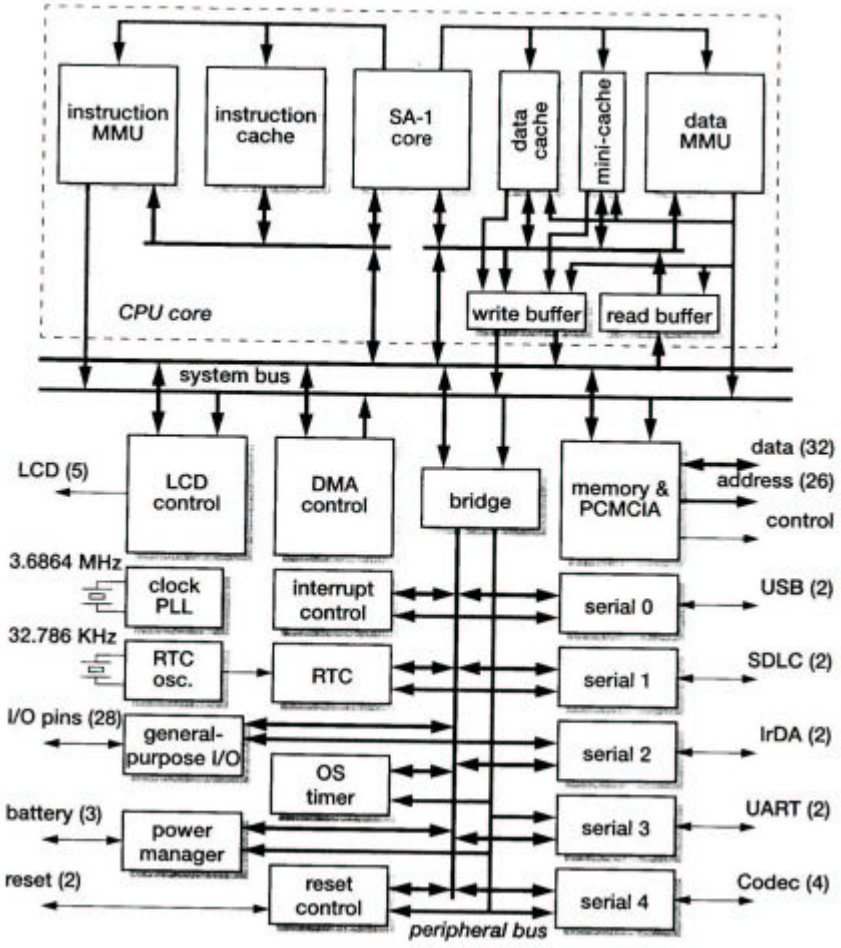


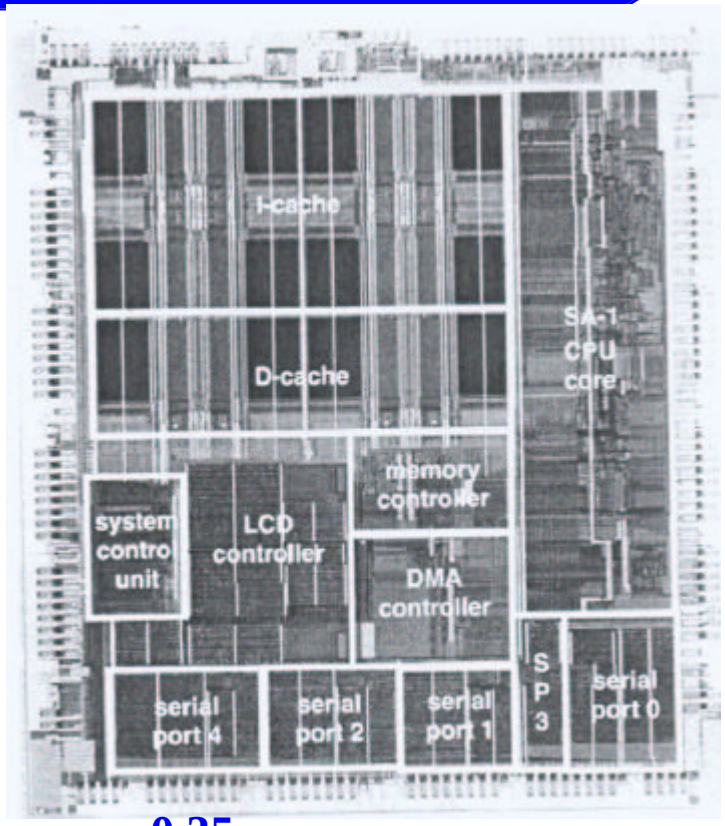


Proceso: 0,6 mm
Transistores: 550.000
Niveles de metalización: 2
Area: 70 mm^2
Potencia: 690 mW
VDD: 5 V
Reloj: 33 MHz

MIPs: 30
MIPs/W: 43

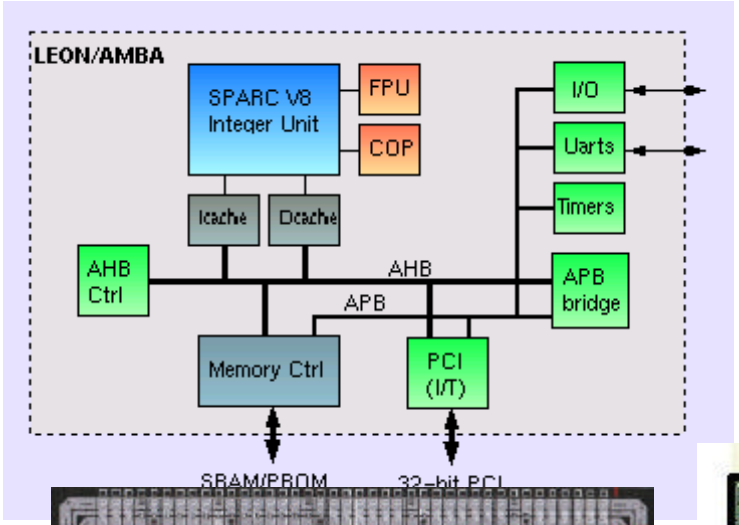
Ejemplo: SA-1100

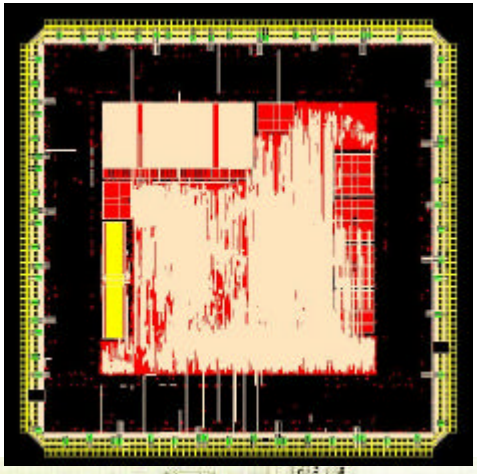


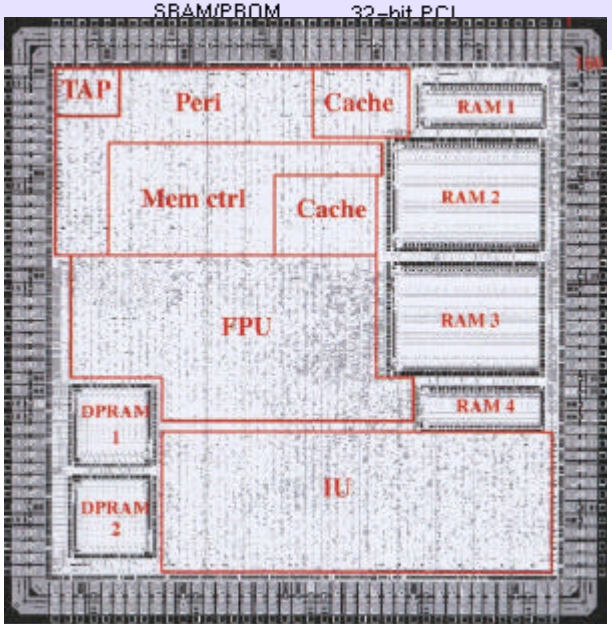


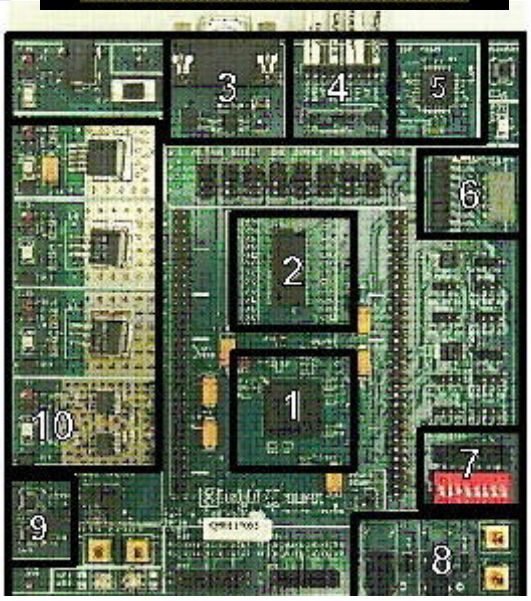
Proceso: 0,35 mm
Transistores: 2.500.000
Niveles de metalización: 32
Area: 75 mm^2
Potencia: 330/550 mW **MIPs: 220/250**
VDD: 1,5/2 V **MIPs/W: 665/450**
Reloj: 190/220 MHz

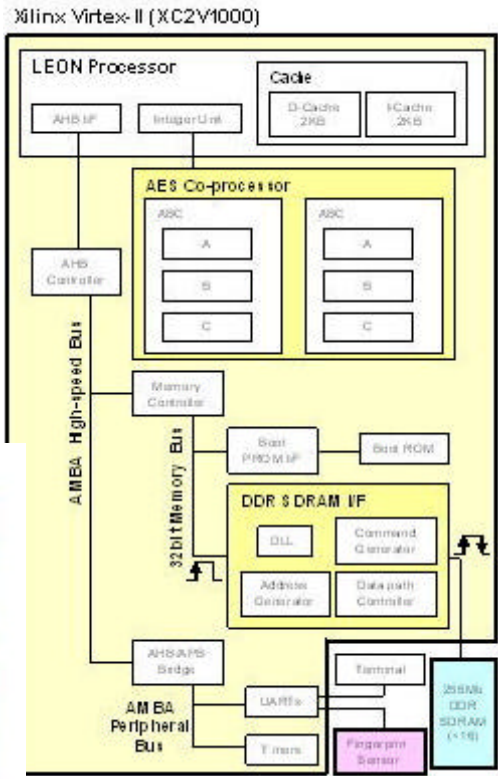
Ejemplo: LEON e implementaciones



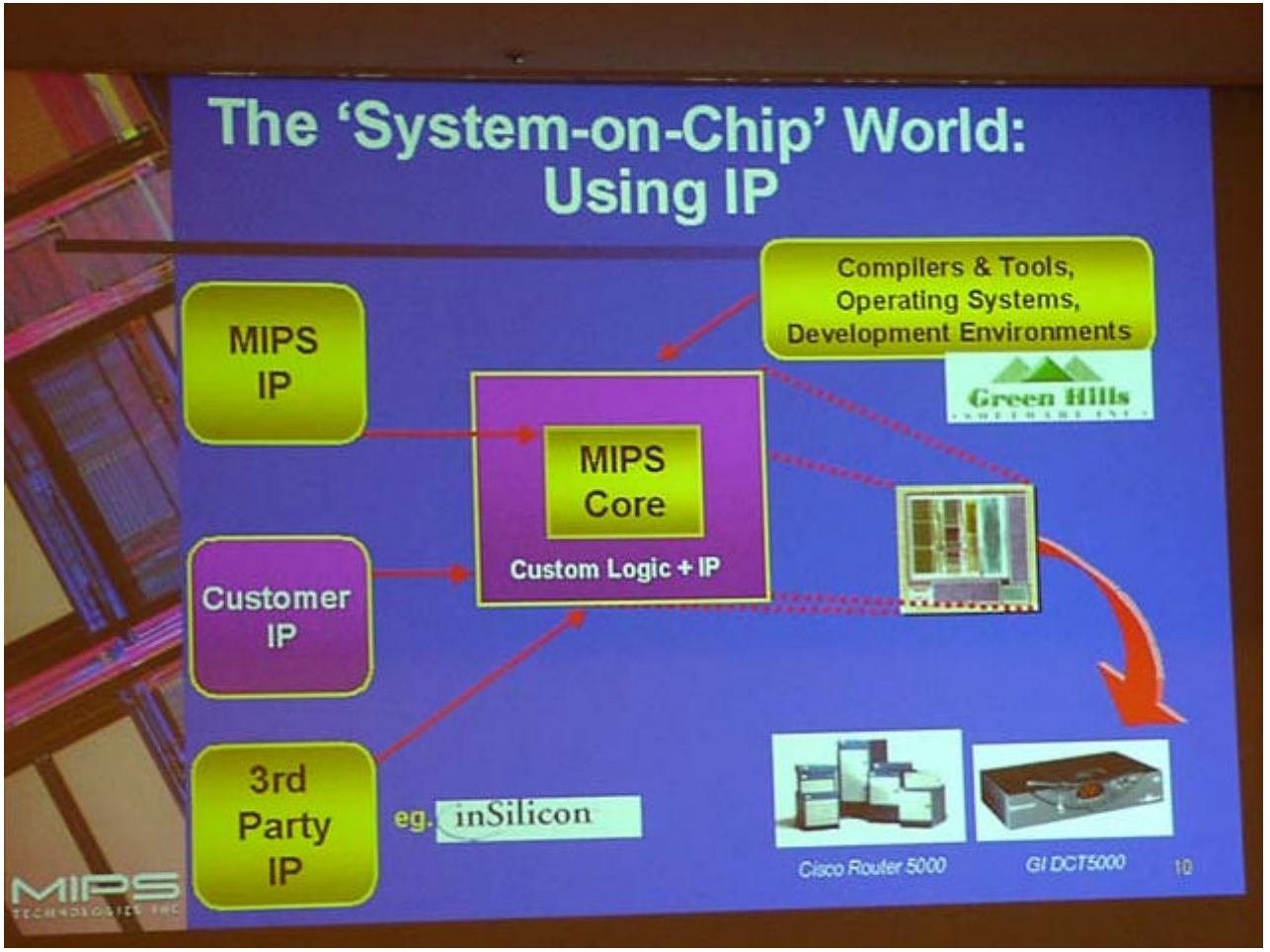




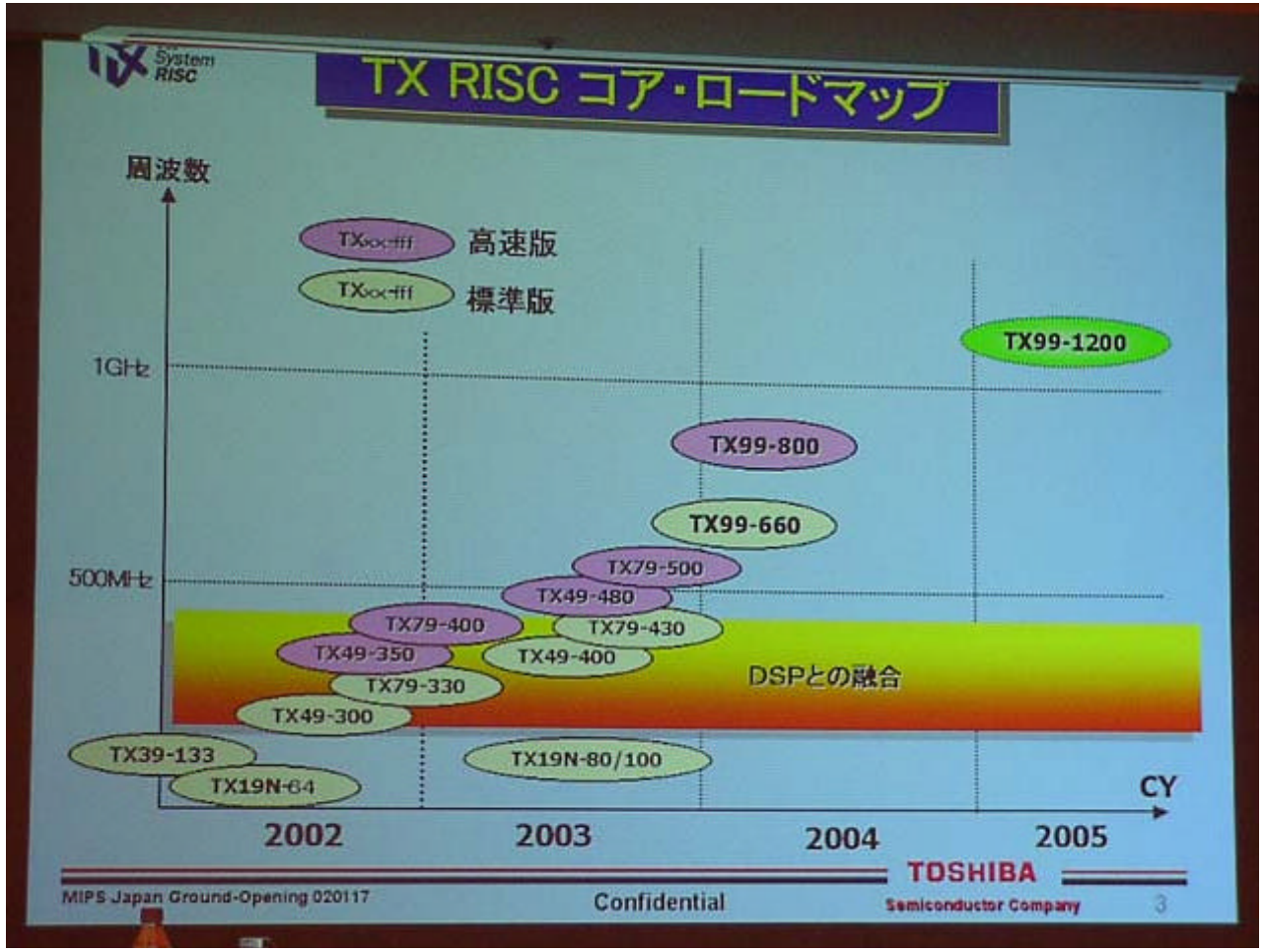


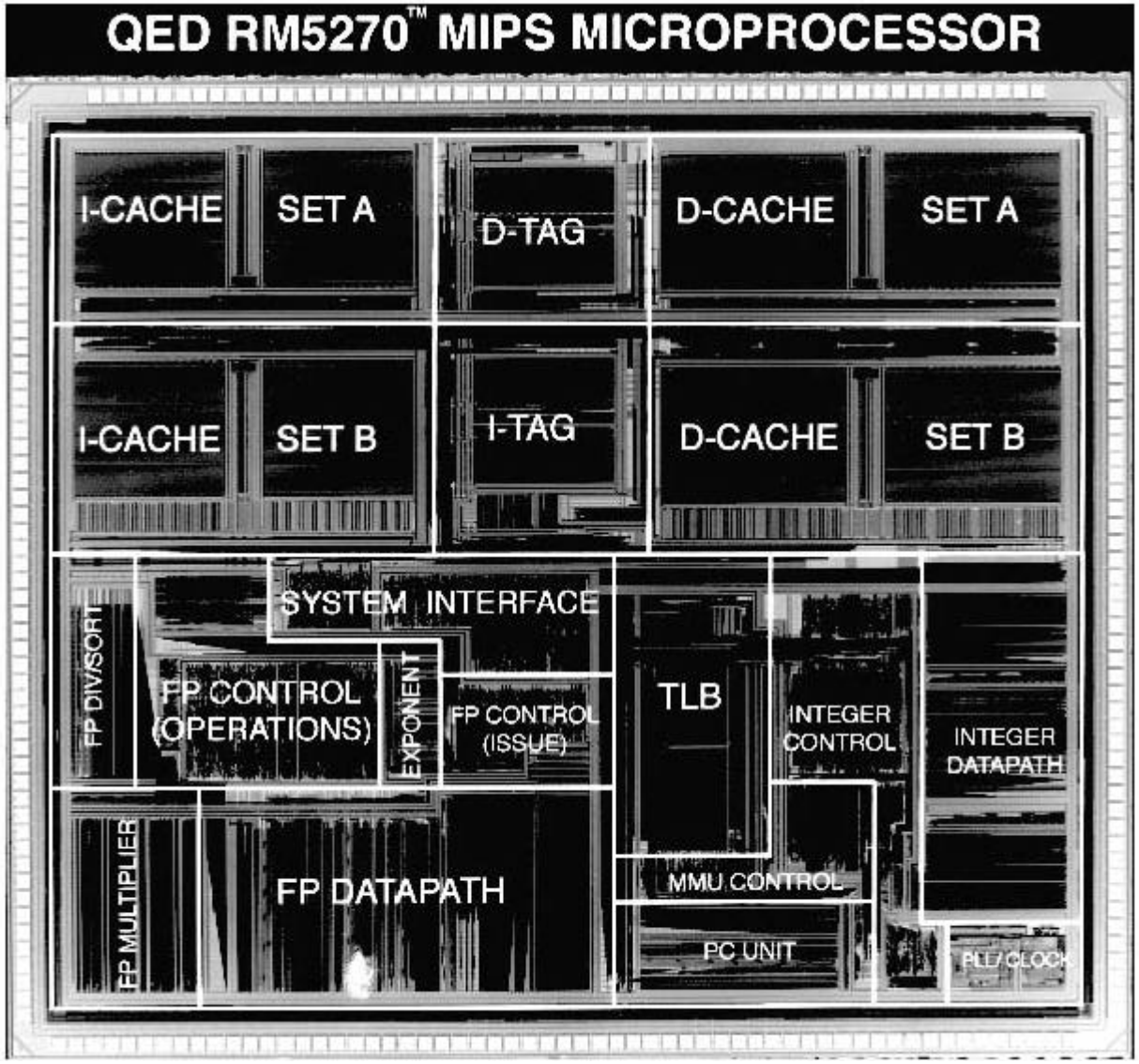


Evolución actual y en el futuro cercano

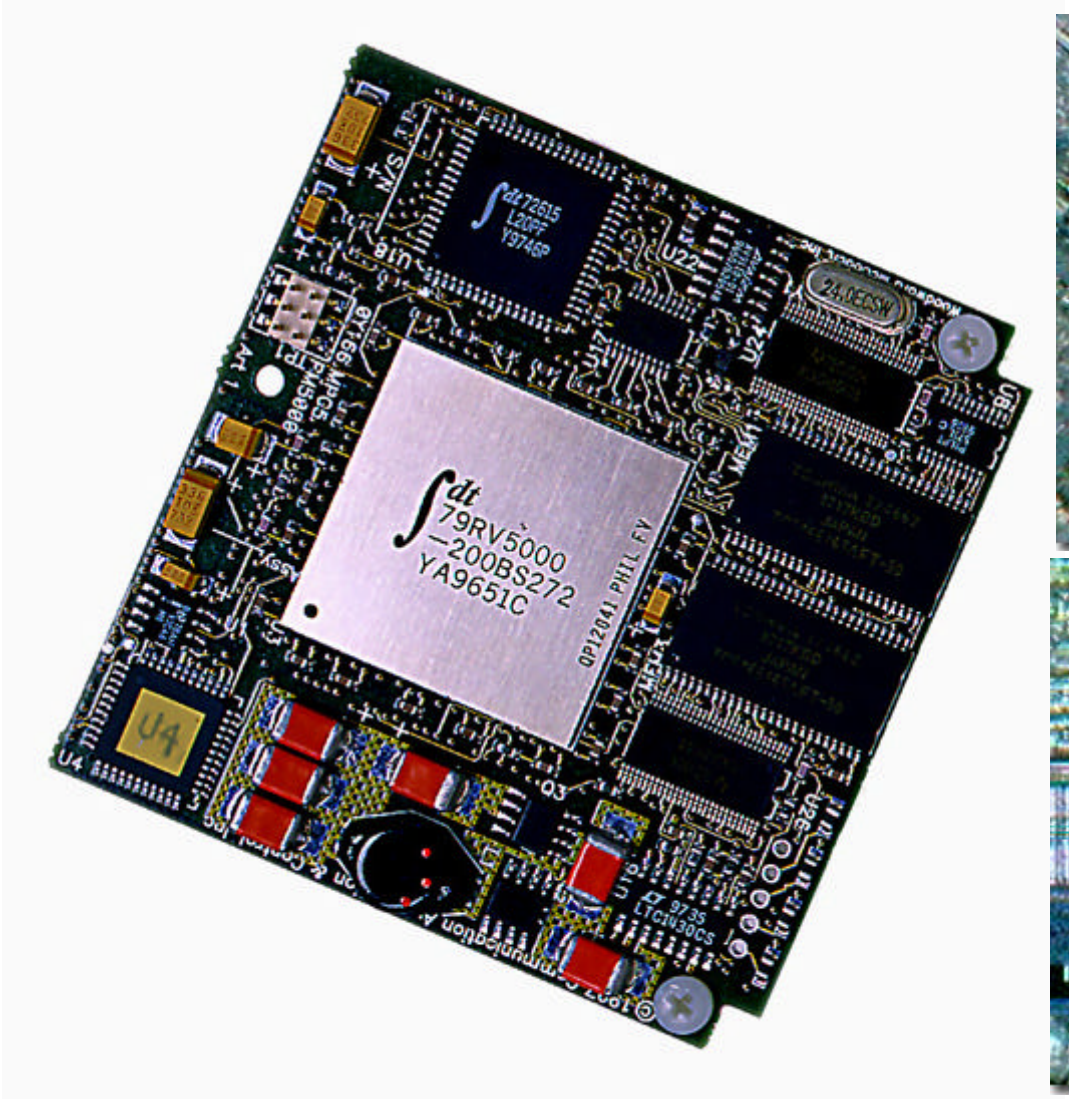


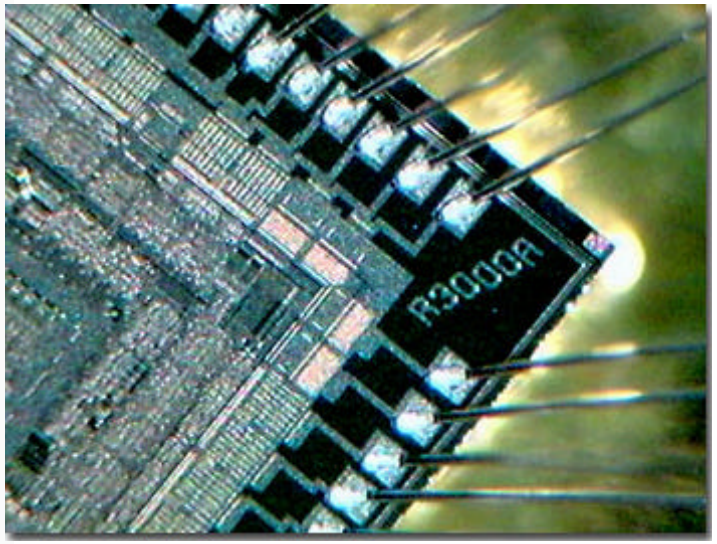
Evolución actual y en el futuro cercano

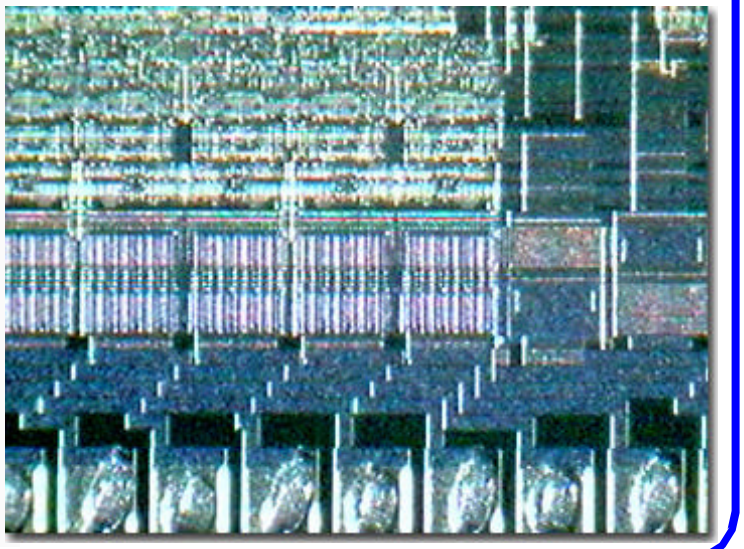


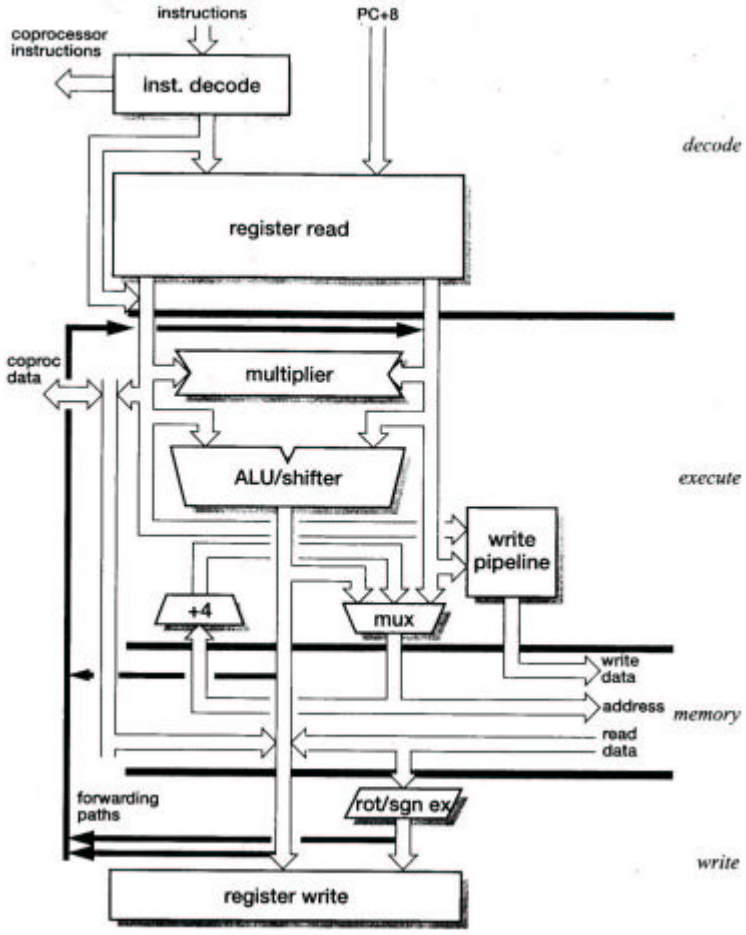

QED RM5270™ MIPS MICROPROCESSOR
I-CACHE
SET A
D-TAG
D-CACHE
SET A
I-CACHE
SET B
I-TAG
D-CACHE
SET B
SYSTEM INTERFACE
FP DIV/SQRT
FP CONTROL (OPERATIONS)
EXPONENT
FP CONTROL (ISSUE)
TLB
INTEGER CONTROL
INTEGER DATAPATH
FP MULTIPLIER
FP DATAPATH
MMU CONTROL
PC UNIT
PLL/CLOCK

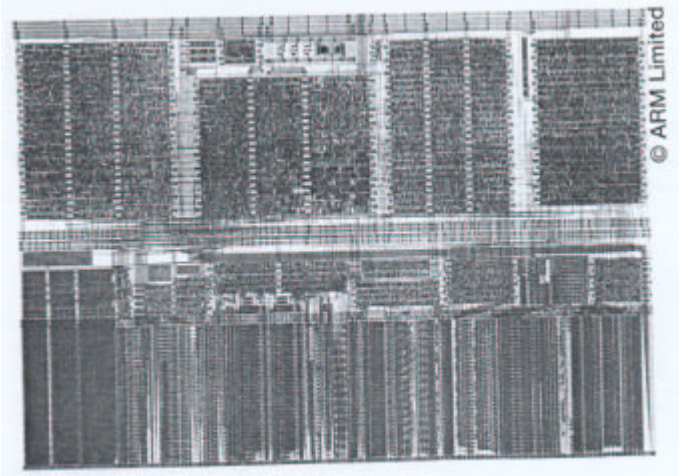
Evolución actual y en el futuro cercano











Proceso: 0,25 mm
Transistores: 110.000
Niveles de metalización: 3
Area: 2,1 mm^2
Potencia: 150 mW
VDD: 2,5 V
Reloj: 200 MHz

MIPs: 220
MIPs/W: 1500