Introducción a las Arquitecturas Modernas

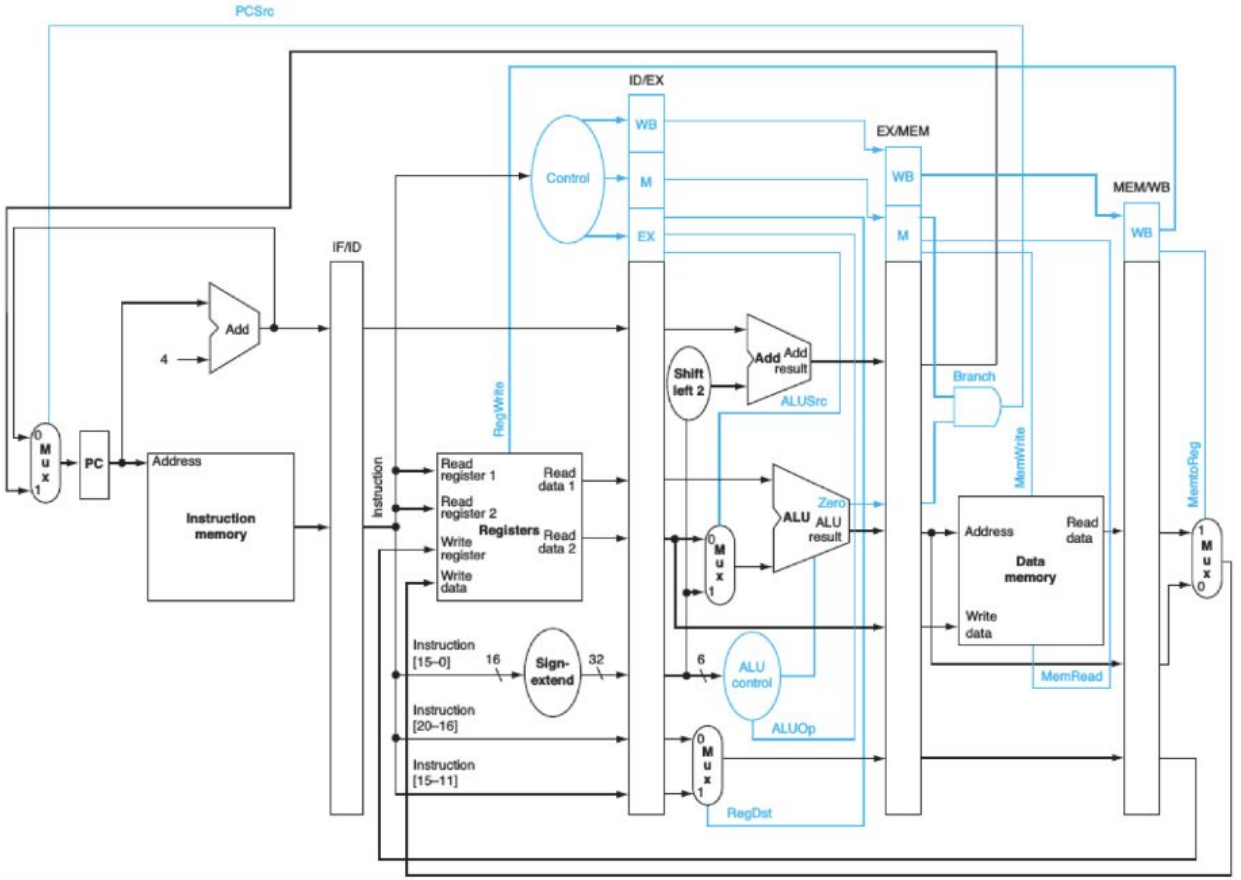
- Paralelismo a nivel de instrucciones
  - Procesadores segmentados (pipeline) 80’
  - Ejecución múltiple (multiple issue):
    - Superescalares (planificación dinámica) 90’,
    - VLIW (planificacióni estática)
  - Rendimiento

- Paralelismo a nivel de procesadores
  - Arquitecturas multiprocesador (2004),
  - Arquitecturas multicomputador (2010).

- Mix de unidades de procesamiento con arquitecturas diferentes
  - Arquitecturas específicas para ciertas aplicaciones o cálculo

Introducción a las Arquitecturas Modernas

- Paralelismo a nivel de instrucciones:
  - Procesadores segmentados (pipeline),



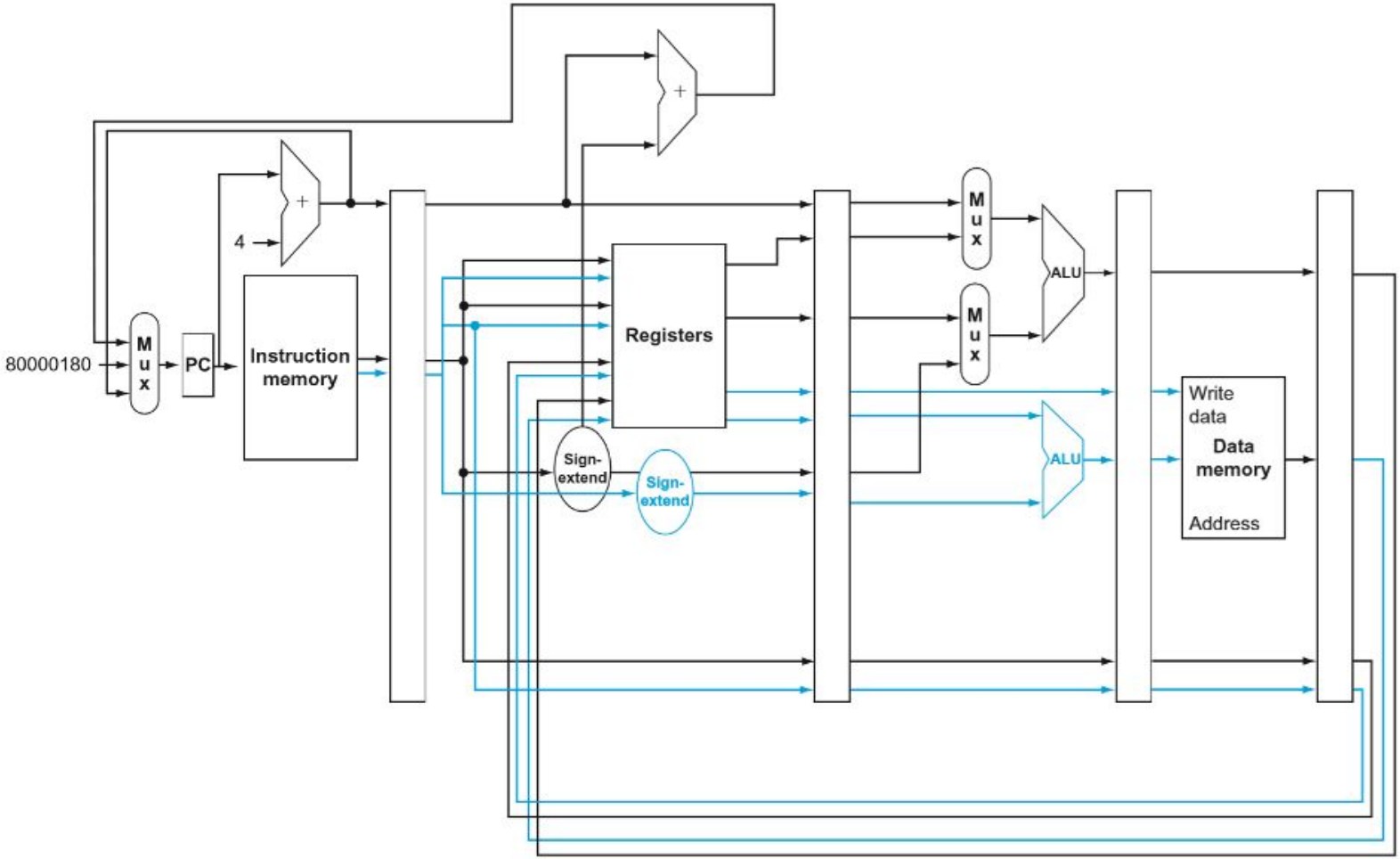


FIGURE 4.69 A static two-issue datapath. The additions needed for double issue are highlighted: another 32 bits from instruction

¿Qué UNIDADES se deben modificar? (piense tambien en las que no están presentes en el diseño)
¿Qué más hace falta para aprovechar esta microarquitectura?

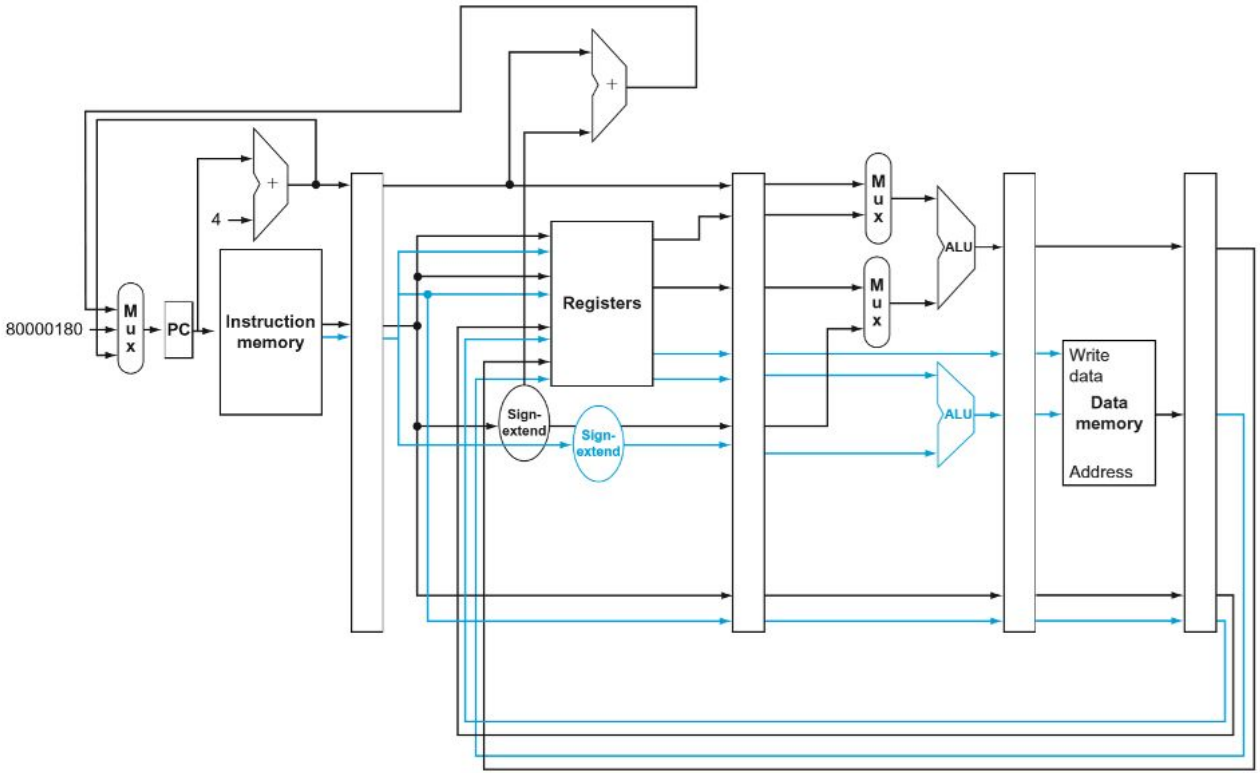


FIGURE 4.69 A static two-issue datapath. The additions needed for double issue are highlighted: another 32 bits from instruction

¿Qué UNIDADES se deben modificar? (piense tambien en las que no están presentes en el diseño)
¿Qué más hace falta para aprovechar esta microarquitectura?

Suponga este programa:

```
lw $t1, 0($t0)
lw $t2, 4($t0)
lw $t3, 8($t0)
lw $t4, 12($t0)
lw $t5, 16($t0)
add $s3, $t1, $t1
sub $s4, $t2, $t2
add $s5, $t3, $t3
and $s6, $t4, $t4
```

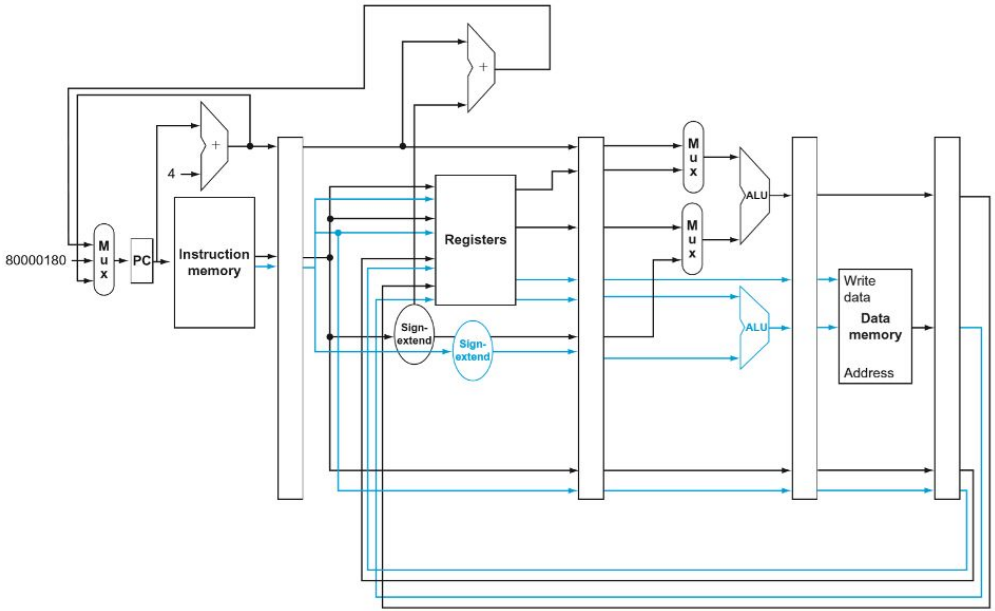


FIGURE 4.69 A static two-issue datapath. The additions needed for double issue are highlighted: another 32 bits from instruction

¿Qué UNIDADES se deben modificar? (piense tambien en las que no están presentes en el diseño)
¿Qué más hace falta para aprovechar esta microarquitectura?

Suponga este programa:

```
lw $t1, 0($t0)
lw $t2, 4($t0)
lw $t3, 8($t0)
lw $t4, 12($t0)
lw $t5, 16($t0)
add $s3, $t1, $t1
sub $s4, $t2, $t2
add $s5, $t3, $t3
and $s6, $t4, $t4
```

Rendimiento
MIPS clasico:

9 ciclos
(si las cachés no ofrecen ciclos extras para los lw)

Versión del compilador para esta microarquitectura:

```
nop
lw $t1, 0($t0)
add $s3, $t1, $t1
lw $t2, 4($t0)
sub $s4, $t2, $t2
lw $t3, 8($t0)
add $s5, $t3, $t3
lw $t4, 12($t0)
and $s6, $t4, $t4
lw $t5, 16($t0)
```

Rendimiento
Diseño MIPS con ejecución múltiple:

5 ciclos

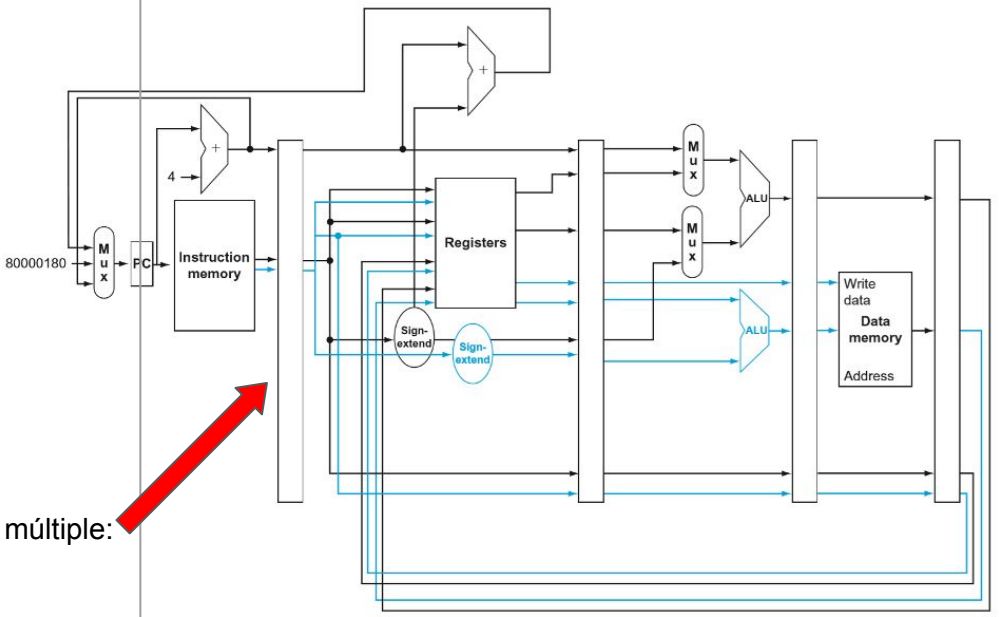


FIGURE 4.69 A static two-issue datapath. The additions needed for double issue are highlighted: another 32 bits from instruction

¿Qué UNIDADES se deben modificar? (piense tambien en las que no están presentes en el diseño)
¿Qué más hace falta para aprovechar esta microarquitectura?

Suponga este programa:

lw $t1, 0($t0)
lw $t2, 4($t0)
lw $t3, 8($t0)
lw $t4, 12($t0)
lw $t5, 16($t0)
add $s3, $t1, $t1
sub $s4, $t2, $t2
add $s5, $t3, $t3
and $s6, $t4, $t4
	nop
	lw $t1, 0($t0)
	add $s3, $t1, $t1
	lw $t2, 4($t0)
	sub $s4, $t2, $t2
	lw $t3, 8($t0)
	add $s5, $t3, $t3
	lw $t4, 12($t0)
	and $s6, $t4, $t4
	lw $t5, 16($t0)

El COMPILADOR debe encargarse (planificación estática)

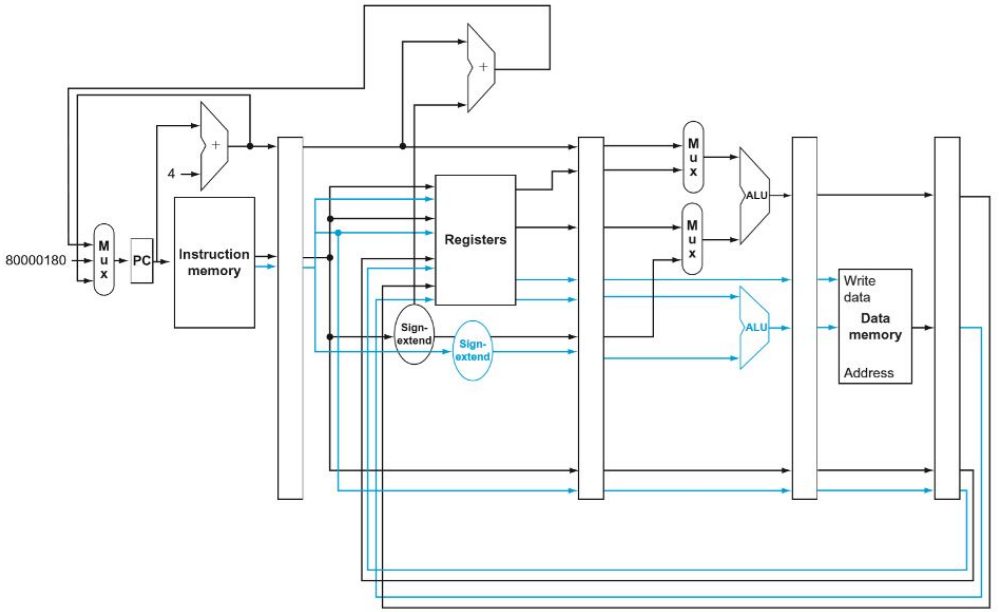


FIGURE 4.69 A static two-issue datapath. The additions needed for double issue are highlighted: another 32 bits from instruction

Ejecución múltiple con planificación estática:

- Tiene sus limitaciones:
  - No es posible sostener el mejor rendimiento de manera sostenida.
  - Si la microarquitectura cambia se debe volver a recompilar.
  - TBC

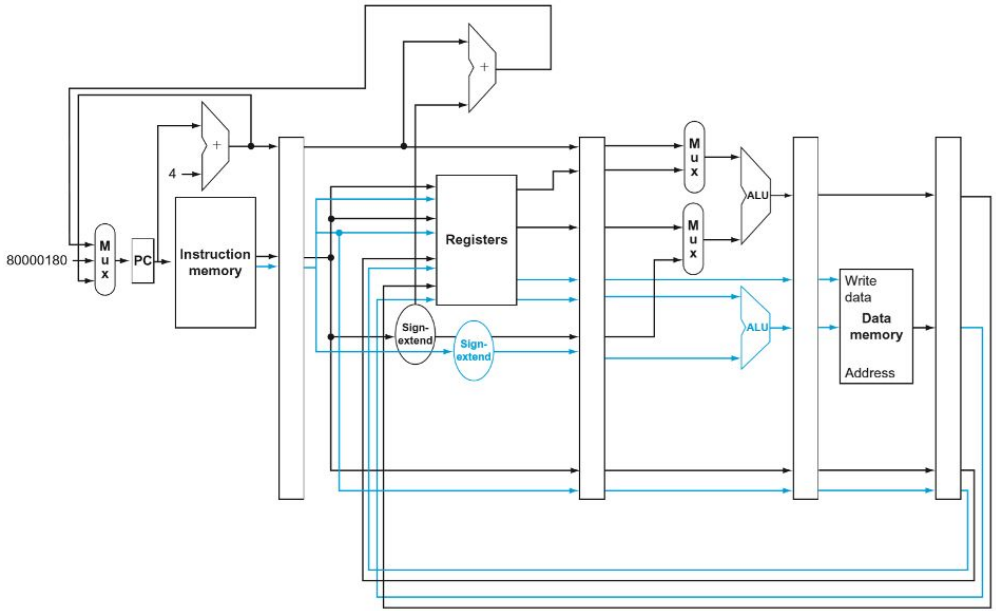
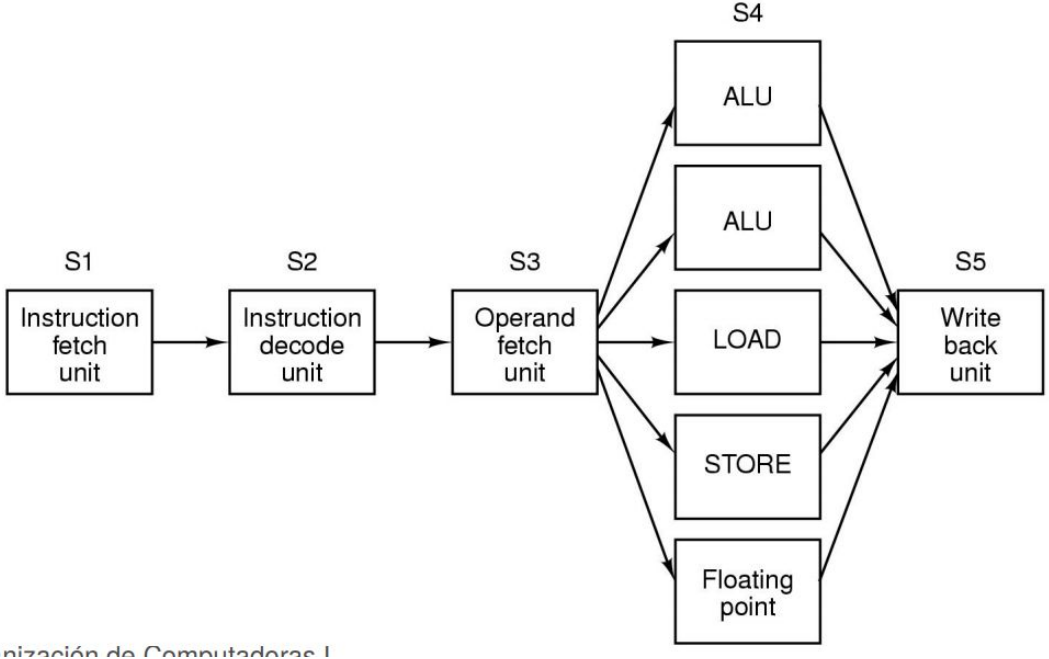


FIGURE 4.69 A static two-issue datapath. The additions needed for double issue are highlighted: another 32 bits from instruction

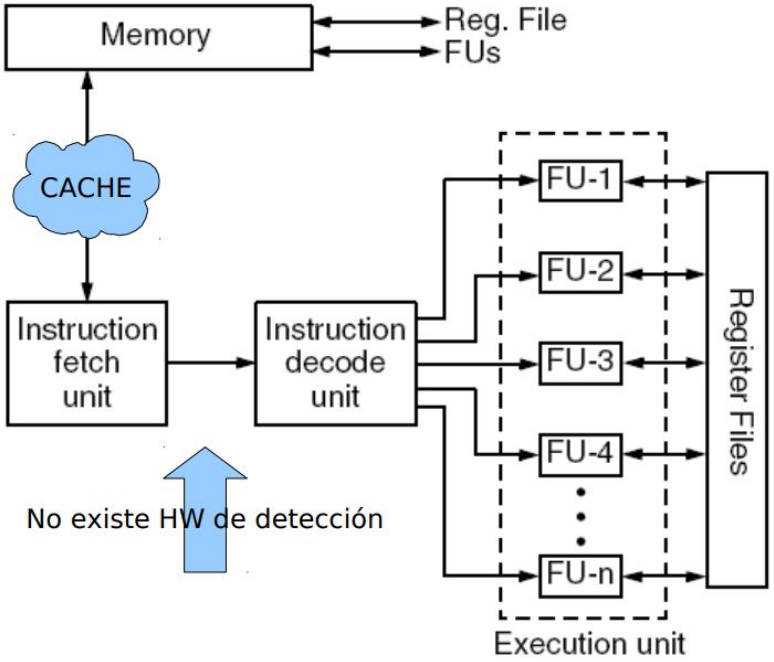
Procesadores de altas prestaciones

- Paralelismo a nivel de instrucciones
  Ejecución múltiple (multiple issue):
  - Superescalares,
  - VLIW



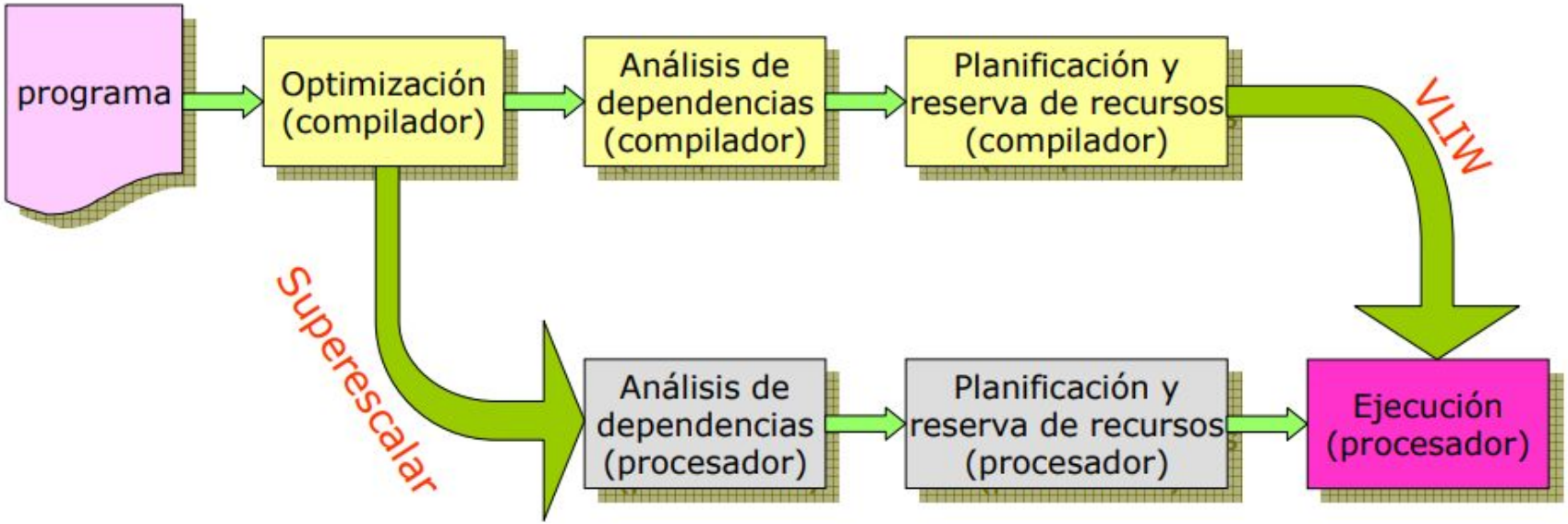
Introducción a las Arquitecturas Modernas

- Paralelismo a nivel de instrucciones
  Ejecución múltiple (multiple issue):
  - VLIW (planificacióni estática),



Introducción a las Arquitecturas Modernas

- Paralelismo a nivel de instrucciones
  Ejecución múltiple (multiple issue):
  - VLIW (planificacióni estática) vs superscalar



Introducción a las Arquitecturas Modernas

- Paralelismo a nivel de instrucciones
  Ejecución múltiple (multiple issue):
  - VLIW (planificacióni estática)

Instr.	load/store & saltos	ALU FX	ALU FP
1	ld f0, 0(r1)	nop	nop
2	ld f0, 0(r1-8)	nop	nop
3	ld f0, 0(r1-16)	nop	addd f4, f0, f2
4	ld f0, 0(r1-24)	nop	addd f8, f6, f2
5	ld f0, 0(r1-32)	nop	addd f12, f10, f2
6	sd f0, 0(r1)	nop	addd f16, f14, f2
7	sd f0, 0(r1-8)	nop	addd f20, f18, f2
8	sd f0, 0(r1-16)	nop	nop
9	sd f0, 0(r1-24)	nop	nop
10	sd f0, 0(r1-32)	subi r1, r1, #8	nop
11	nop	nop	nop
12	bnez r1, lazo	nop	nop
13	nop	nop	nop

Introducción a las Arquitecturas Modernas

- Paralelismo a nivel de instrucciones
  - Procesadores segmentados (pipeline)
  - Ejecución múltiple (multiple issue):
    - Superescalares (planificación dinámica),
    - VLIW (planificacióni estática)

¿Qué mejora la segmentación?

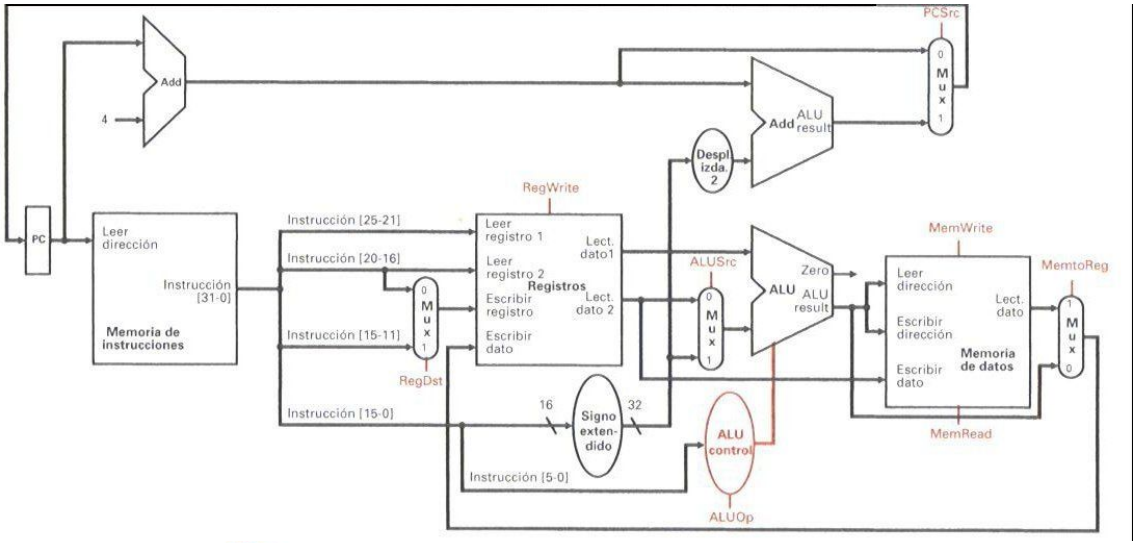
Tiempo de ejecución = Número de instrucciones × CPI × Tiempo de ciclo

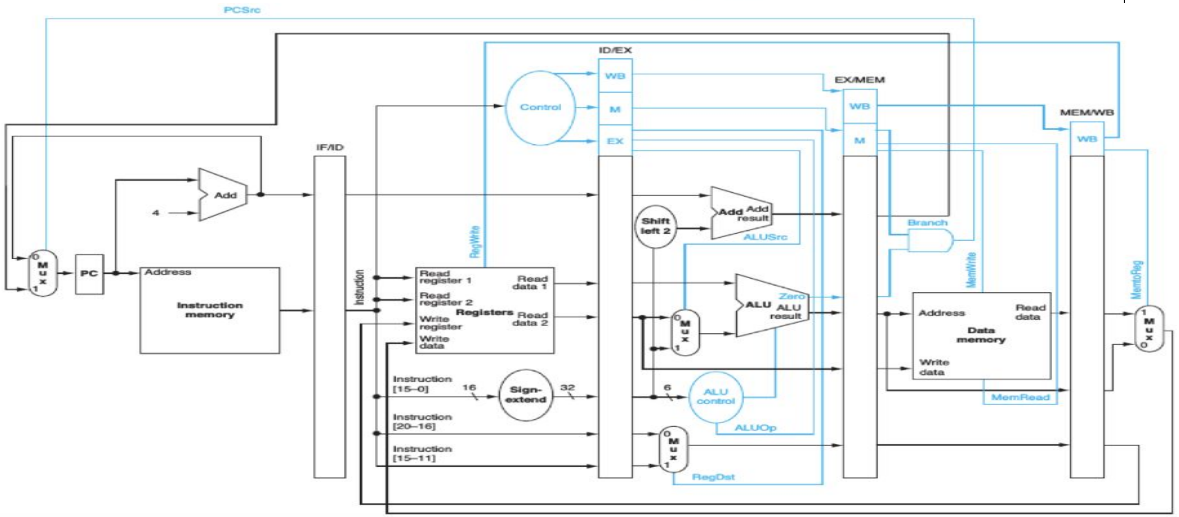
o bien, dado que la frecuencia es el inverso del tiempo de ciclo:

$$\text{Tiempo de ejecución} = \frac{\text{Número de instrucciones} \times \text{CPI}}{\text{Frecuencia de reloj}}$$

Estas fórmulas son especialmente útiles porque distinguen los tres factores claves que influyen en las prestaciones. Estas fórmulas se pueden utilizar para comparar dos realizaciones diferentes o para evaluar un diseño alternativo si se conoce el impacto en estos tres parámetros.

Componentes de las prestaciones	Unidades de medida
Tiempo de ejecución de CPU de un programa	Segundos por programa
Número de instrucciones	Número de instrucciones ejecutadas por el programa
Ciclos por instrucción (CPI)	Número medio de ciclos por instrucción
Tiempo de ciclo del reloj	Segundos por ciclo de reloj


PCSrc
Add
4
Add
ALU result
Desplizda 2
RegWrite
PC
Leer dirección
Instrucción [31-0]
Memoria de instrucciones
Instrucción [25-21]
Instrucción [20-16]
Instrucción [15-11]
Instrucción [15-0]
Instrucción [5-0]
Leer registro 1
Leer registro 2
Registros
Escribir registro
Escribir dato
Lect. dato1
Lect. dato 2
RegDst
ALUSrc
ALU
Zero
ALU result
MemWrite
MemtoReg
Leer dirección
Escribir dirección
Escribir dato
Lect. dato
Memoria de datos
MemRead
16
Signo extendido
32
ALU control
ALUOp
M u x


PCSrc
ID/EX
EX/MEM
MEM/WB
WB
M
EX
Control
IF/ID
Add
4
PC
Address
Instruction memory
Instruction
RegWrite
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
Instruction [15–0]
Instruction [20–16]
Instruction [15–11]
16
Sign-extend
32
Shift left 2
Add Add result
ALUSrc
ALU Zero ALU result
6
ALU control
ALUOp
RegDst
Branch
MemWrite
MemRead
MemtoReg
Address
Data memory
Read data
Write data

Introducción a las Arquitecturas Modernas

- Paralelismo a nivel de instrucciones
  - Procesadores segmentados (pipeline)
  - Ejecución múltiple (multiple issue):
    - Superescalares (planificación dinámica),
    - VLIW (planificacióni estática)

¿Qué mejora la ejecución múltiple?

$$\text{Tiempo de ejecución} = \text{Número de instrucciones} \times \text{CPI} \times \text{Tiempo de ciclo}$$

o bien, dado que la frecuencia es el inverso del tiempo de ciclo:

$$\text{Tiempo de ejecución} = \frac{\text{Número de instrucciones} \times \text{CPI}}{\text{Frecuencia de reloj}}$$

Estas fórmulas son especialmente útiles porque distinguen los tres factores claves que influyen en las prestaciones. Estas fórmulas se pueden utilizar para comparar dos realizaciones diferentes o para evaluar un diseño alternativo si se conoce el impacto en estos tres parámetros.

Componentes de las prestaciones	Unidades de medida
Tiempo de ejecución de CPU de un programa	Segundos por programa
Número de instrucciones	Número de instrucciones ejecutadas por el programa
Ciclos por instrucción (CPI)	Número medio de ciclos por instrucción
Tiempo de ciclo del reloj	Segundos por ciclo de reloj

Introducción a las Arquitecturas Modernas

- Paralelismo a nivel de instrucciones
  - Procesadores segmentados (pipeline)
  - Ejecución múltiple (multiple issue):
    - Superescalares (planificación dinámica),
    - VLIW (planificacióni estática)

¿Qué mejora la ejecución múltiple?

CPI se puede reescribir como $\frac{1}{IPC}$

Tiempo de ejecución = Número de instrucciones × CPI × Tiempo de ciclo

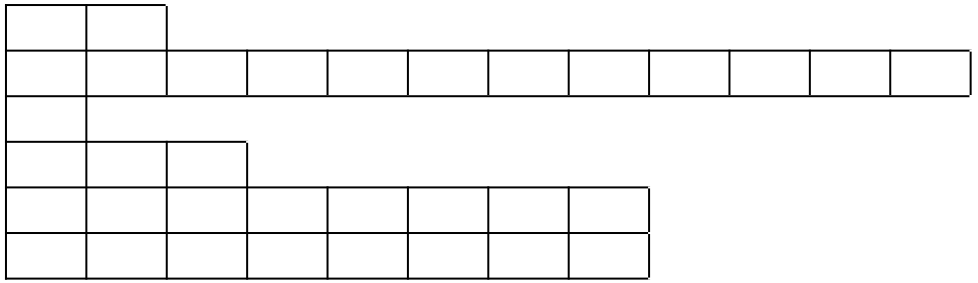
o bien, dado que la frecuencia es el inverso del tiempo de ciclo:

$$\text{Tiempo de ejecución} = \frac{\text{Número de instrucciones} \times \text{CPI}}{\text{Frecuencia de reloj}}$$

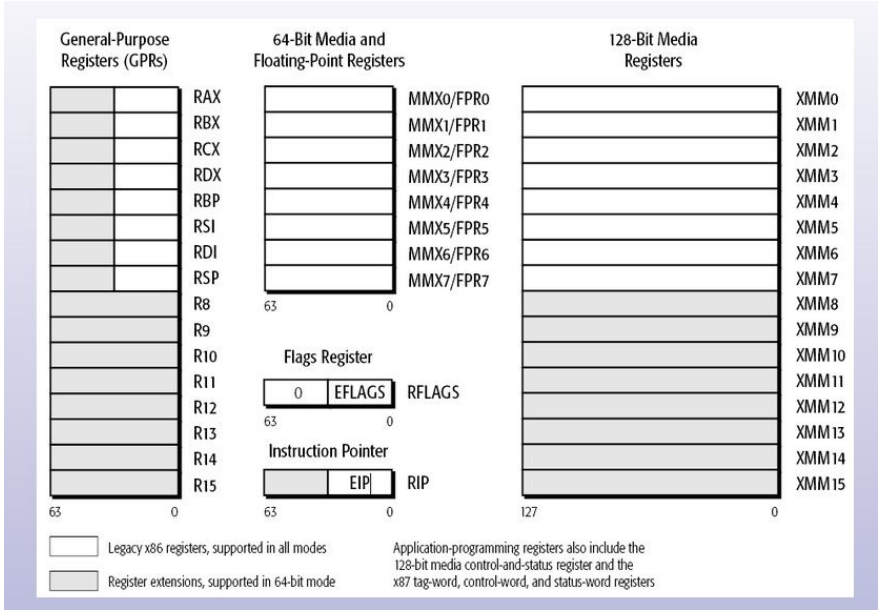
Estas fórmulas son especialmente útiles porque distinguen los tres factores claves que influyen en las prestaciones. Estas fórmulas se pueden utilizar para comparar dos realizaciones diferentes o para evaluar un diseño alternativo si se conoce el impacto en estos tres parámetros.

Componentes de las prestaciones	Unidades de medida
Tiempo de ejecución de CPU de un programa	Segundos por programa
Número de instrucciones	Número de instrucciones ejecutadas por el programa
Ciclos por instrucción (CPI)	Número medio de ciclos por instrucción
Tiempo de ciclo del reloj	Segundos por ciclo de reloj

Introducción a las Arquitecturas Modernas AMD64 (Intel y AMD de 64bits, extensión de x86 de 32bits)



- Instrucciones de tamaño variable
- Pocos registros (no ahora)
- Muchos modos de direccionamiento
- Conjunto de instrucciones amplio y complejo
- Operaciones con 2 operandos (uno es destino)



Introducción a las Arquitecturas Modernas AMD64 (Intel y AMD de 64bits, extensión de x86 de 32bits)

Referencia de algunas instrucciones del Lenguaje Ensamblador

```
mov eax, [ebx]          ; Move the 4 bytes in memory at the address contained in EBX into EAX
mov [var], ebx          ; Move the contents of EBX into the 4 bytes at memory address var. (Note, var is a 32-bit constant).
mov eax, [esi-4]        ; Move 4 bytes at memory address ESI + (-4) into EAX
mov [esi+eax], cl       ; Move the contents of CL into the byte at address ESI+EAX
mov edx, [esi+4*ebx]            ; Move the 4 bytes of data at address ESI+4*EBX into EDX
```

<reg32>	Any 32-bit register (EAX, EBX, ECX, EDX, ESI, EDI, ESP, or EBP)
<reg16>	Any 16-bit register (AX, BX, CX, or DX)
<reg8>	Any 8-bit register (AH, BH, CH, DH, AL, BL, CL, or DL)
<reg>	Any register
<mem>	A memory address (e.g., [eax], [var + 4], or dword ptr [eax+ebx])
<con32>	Any 32-bit constant
<con16>	Any 16-bit constant
<con8>	Any 8-bit constant
<con>	Any 8-, 16-, or 32-bit constant

add eax, 10 — EAX ← EAX + 10
add BYTE PTR [var], 10 — add 10 to the single byte stored at memory address var

sub al, ah — AL ← AL - AH
sub eax, 216 — subtract 216 from the value stored in EAX

imul eax, [var] — multiply the contents of EAX by the 32-bit contents of the memory location var. Store the result in EAX.
imul esi, edi, 25 — ESI → EDI * 25

Introducción a las Arquitecturas Modernas AMD64 (Intel y AMD de 64bits, extensión de x86 de 32bits)

- Superscalar — varias unidades funcionales (ej: ALUs)
- Planificación dinámica — todos los procesadores Intel y AMD
- Especulación — predecir el resultado de una operación
- Predicción de saltos

```
for (i=0; i<10; i++)
      a = b + c;
```

```
for:
      mov eax, 0
      add  eax, ebx
      add eax, ecx
      add edx, 1
      cmp edx, 10
      jne for
```

- Ejecución fuera de orden
  (la demora de una instrucción no impide continuar la ejecución)

Introducción a las Arquitecturas Modernas AMD64 (Intel y AMD de 64bits, extensión de x86 de 32bits)

- Superscalar
  - Posibles bloqueos (a ser superados o aprovechados):
    - Fallo de caché
    - Lectura despues de escritura

```
sw $4,  8($6)
lw $5, 16($7)
```

    - Uso de registros

```
add $6, $7, $8
sw $6, ($9)
addi $6, $0, 1
sw $6, ($10)
```

    - Decenas o cientos de situaciones

Introducción a las Arquitecturas Modernas AMD64 (Intel y AMD de 64bits, extensión de x86 de 32bits)

Instrucciones máquina (**macro-instrucciones**)

- Algunas son decodificadas por hardware
- Otras son convertidas a varias **micro-instrucciones** desde una ROM

Ejemplo:

Macroinstrucción (instrucción máquina)		Convertida a al menos 3 micro-instrucciones
add [eax], ebx	:	leer desde la memoria a registro sumar ebx a registro escribir en memoria el resultado

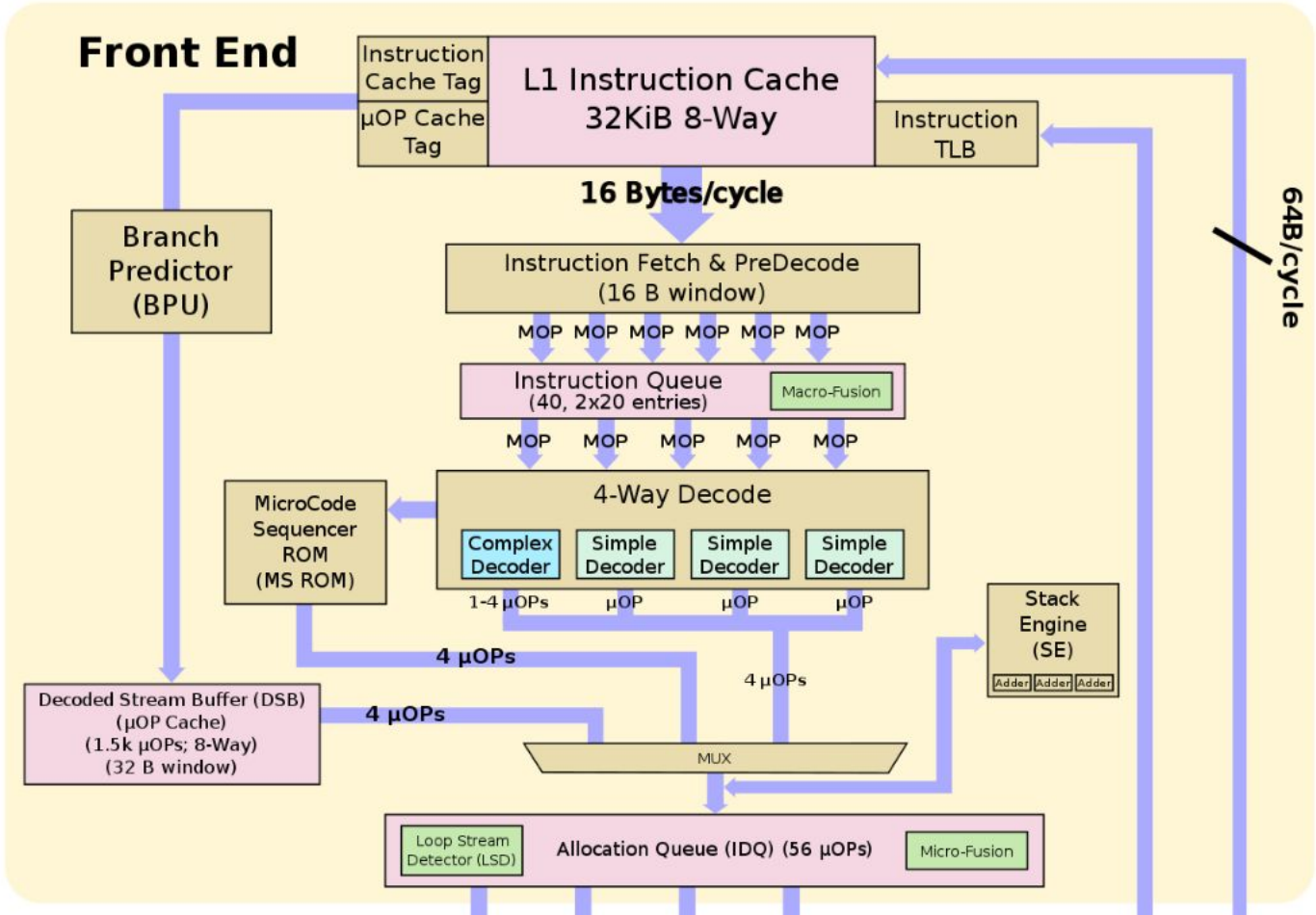
Ejemplo: Microarquitectura Haswell: 4ta generación de procesadores Intel (ej: i7-4790)

Superscalar
(8 unidades funcionales)

Segmentado
(14 a 19 etapas)

Ejecución Fuera de orden
(commit in-order)

Ejecución especulativa
(ej: predecir el resultado
de un salto condicional)



Ejemplo: Microarquitectura Haswell: 4ta generación de procesadores Intel (ej: i7-4790)