Ciclo de instrucción
Fetch-Decode-Execute

Ciclo de instrucción

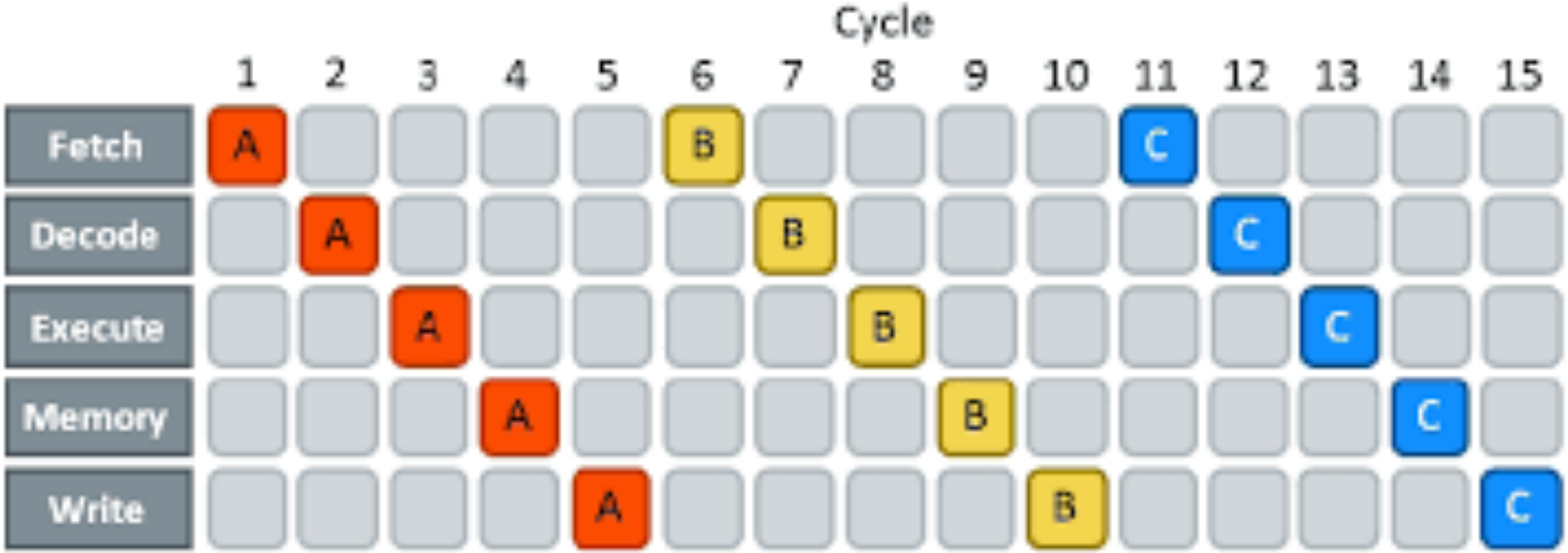
Un **ciclo de instrucción** (también llamado ciclo de fetch-and-execute o ciclo de fetch-decode-execute en inglés) es el período que tarda la unidad central de proceso (CPU) en ejecutar una instrucción de lenguaje máquina.

- Comprende una secuencia de acciones determinadas que debe llevar a cabo la CPU para ejecutar cada instrucción en un programa, cada instrucción del juego de instrucciones de una CPU puede requerir diferente número de ciclos de instrucción para su ejecución.

Un ciclo de instrucción está formado por uno o más ciclos máquina

Ciclo de Instrucción: Fetch

- Cuando están involucrados más de un operando cada uno de ellos requiere un acceso


Cycle
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Fetch
Decode
Execute
Memory
Write
A
B
C

Memoria Interna: Registros

Registros Visibles para el Usuario

- Registros de Propósito General
- Registros de dirección (parcialmente generales, registros índices o Stack Pointer)
- Registros de datos
- Registros de Condición (Flags son fijados por HW)

Control y Estado

- Contador de Programa (PC)
- Registro de Instrucción (IR)
- Registro de dirección de Memoria (MAR)
- Registro Intermedio de Memoria (MBR)

Estos registros tienen gran importancia en la ejecucióndel ciclo de instrucción

2.3.2 Actividad de investigación

- Investigar el proceso para la segmentación de instrucciones.
- Hacer un resumen del proceso

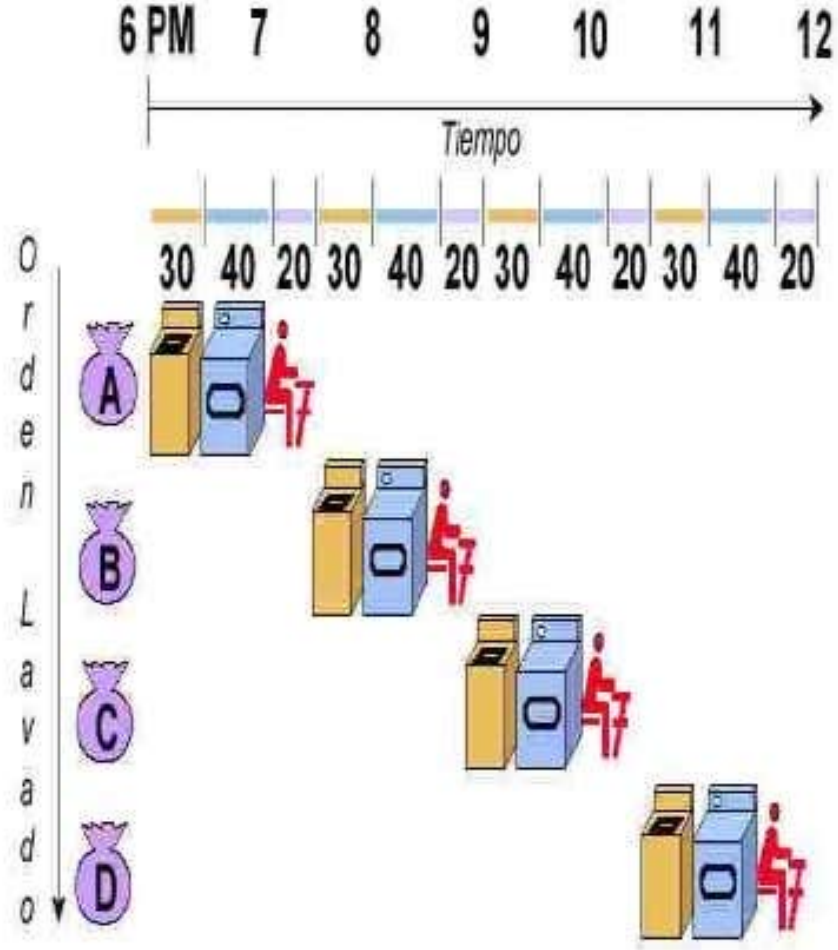
Segmentación de Instrucciones: Pipelining

Instrucciones utilizan los recursos distintos en distintas etapas de la ejecución, entonces se ejecutan múltiples instrucciones simultaneamente siempre y cuando TODAS se encuentren en distintas etapas de ejecución

¿Por qué a la segmentación se le llama Pipelining?

Porque al igual que en una tubería, se aceptan entradas nuevas en un extremo antes de que las anteriores sean salidas en el otro extremo

Segmentación de Instrucciones



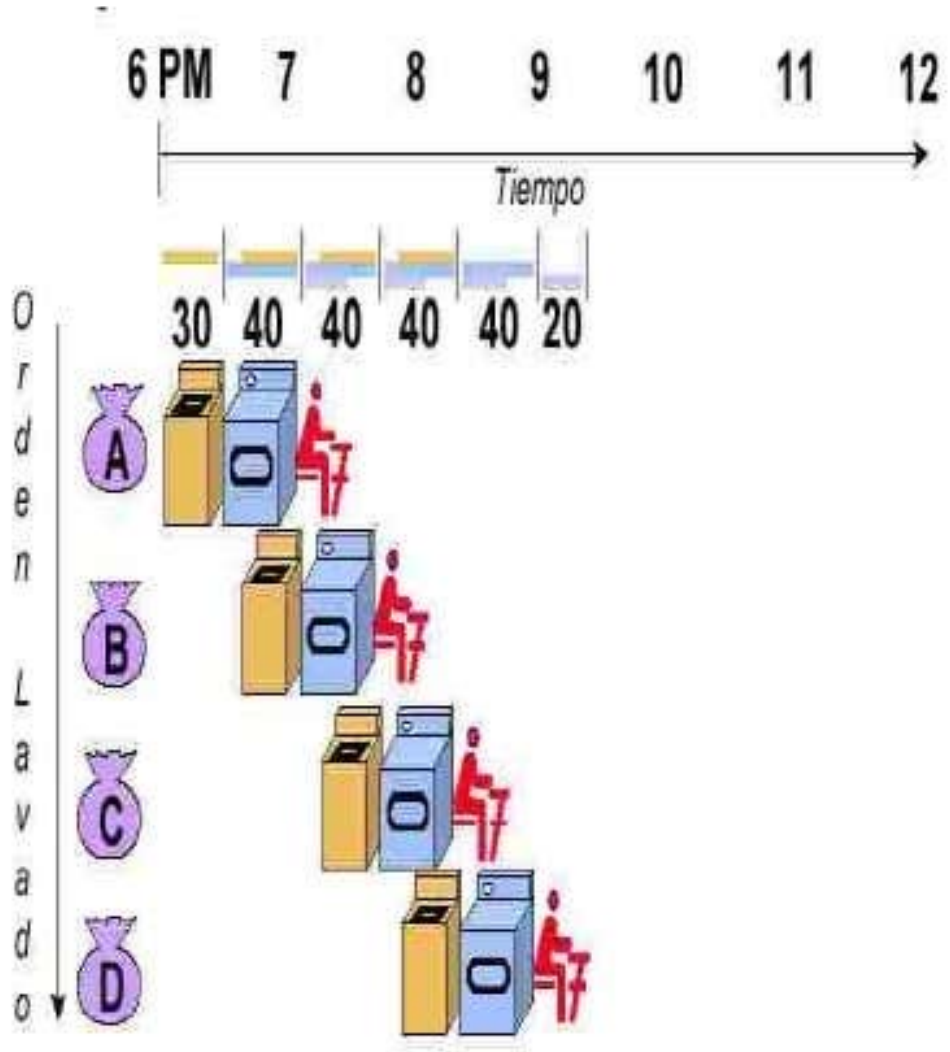
Secuencia de la lavandería

1. Recibir Cargas
2. Lavar Carga i
3. Secar Carga i
4. Planchar Carga i

Las máquinas quedan desocupadas en algunos ciclos

Total (4 cargas) = 6 horas

Segmentación de instrucciones



Lavandería Segmentada

1. Recibir Cargas
2. Lavar Carga i
3. Secar Carga i y lavar Carga i+1
4. Planchar Carga i, secar Carga i+1 y lavar Carga i+2

Total (4 cargas)= 3.5 horas

Segmentación de instrucciones

¿Cómo se ejecuta la segmentación?

En cada ciclo se inicia la ejecución de una instrucción

Existen múltiples instrucciones en ejecución

Se inicia una ejecución en cada ciclo de reloj

La segmentación:

No mejora latencia individual

Mejora el throughput global

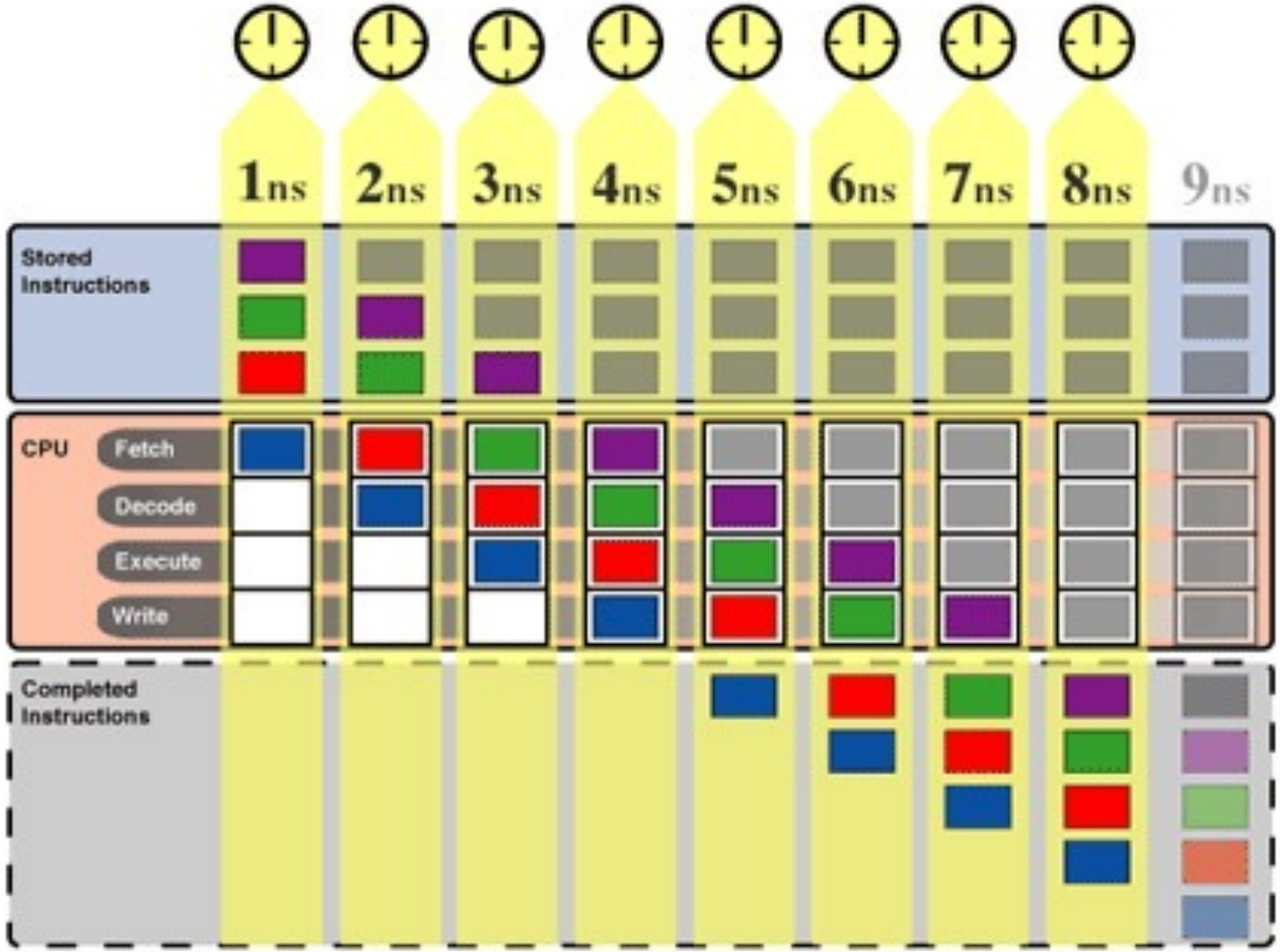
Está limitada por la instrucción más lenta

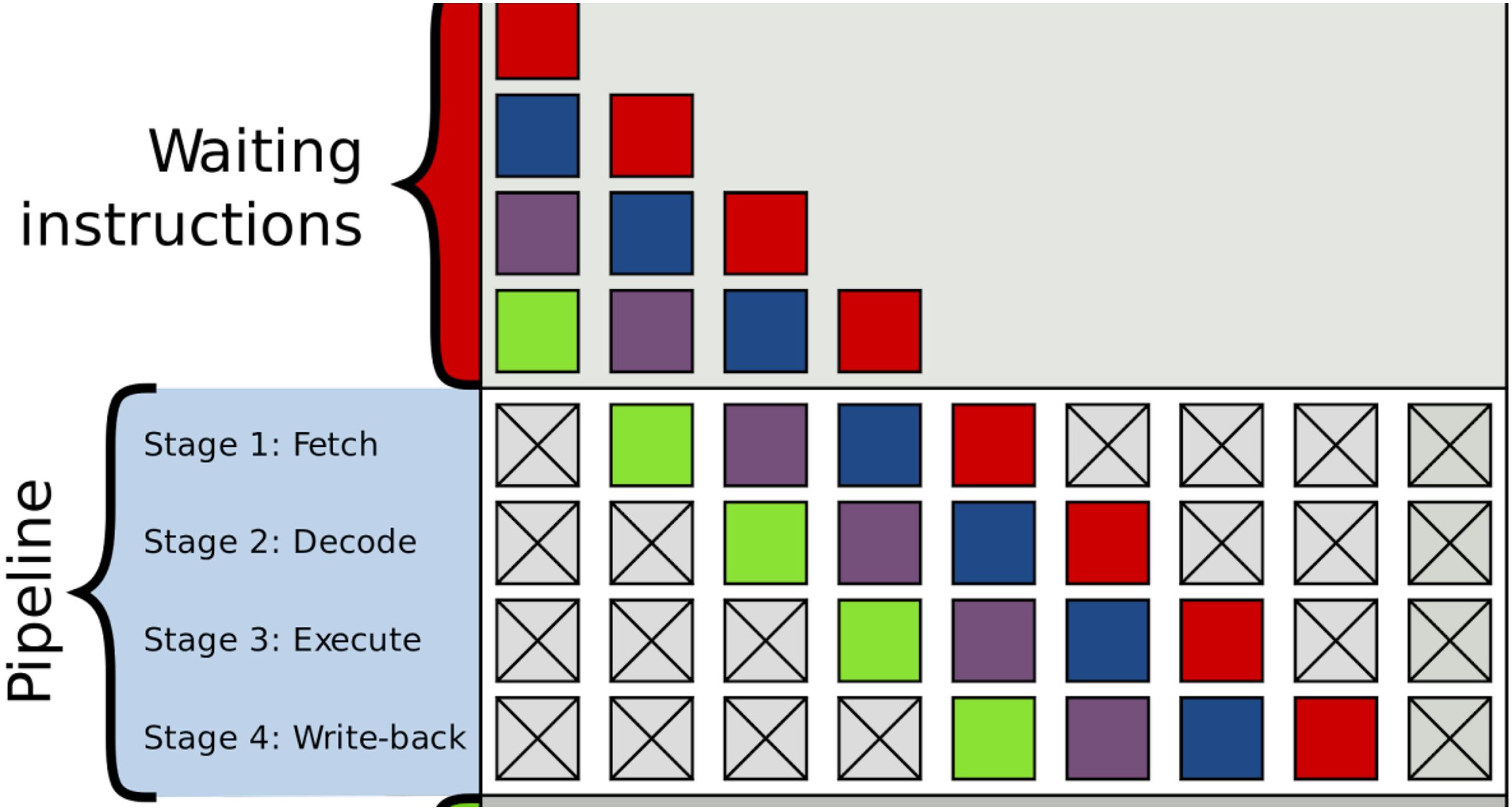
Segmentación de instrucciones

Se descomponen las instrucciones para evitar el desbalanceamiento:

- Buscar instrucción (FI)
- Decodifcar Instrucción (DI)
- Calcular Operandos (CO)
- Buscar Operandos (FO)
- Ejecutar instrucción (EI)
- Escribir Operando (WO)

Como se ve en la figura, el tiempo de ejecución se reduce de 54 a 14 unidades de tiempo


1ns
2ns
3ns
4ns
5ns
6ns
7ns
8ns
9ns
Stored Instructions
CPU
Fetch
Decode
Execute
Write
Completed Instructions


Waiting instructions
Pipeline
Stage 1: Fetch
Stage 2: Decode
Stage 3: Execute
Stage 4: Write-back

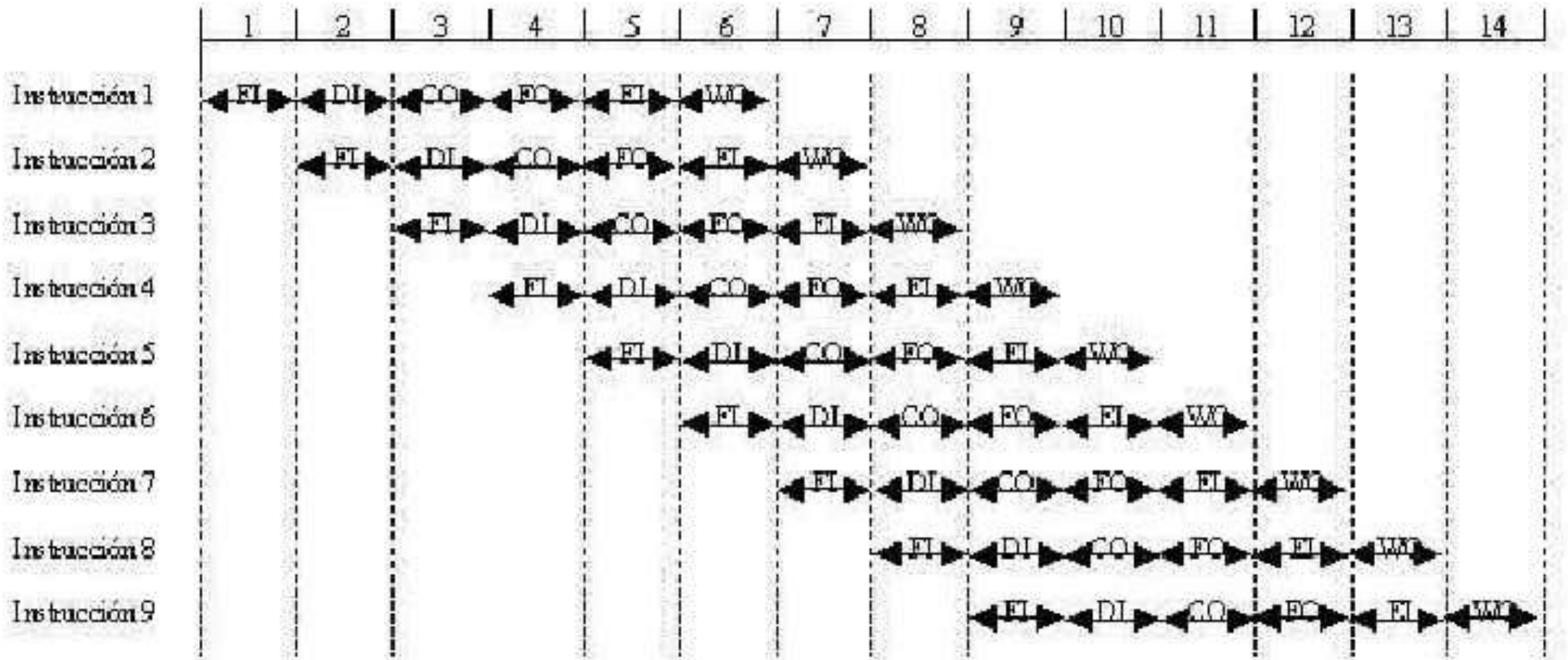


Figura 5.3 Diagrama de tiempos del funcionamiento de cauce de instrucciones

Conjunto de instrucciones: Características y funciones

Actividad. Realizar un listado de las instrucciones del microprocesador 8086

RESUMEN DEL REPERTORIO DE INSTRUCCIONES 8086

GRUPO	INSTRUCCIÓN	DESCRIPCIÓN
TRANSFERENCIA	MOV	Genérica
	MOVSB, MOVSW	Movimiento de cadenas
	LODSB, LODSW	Carga en acum. (cadenas)
	STOSB, STOSW	Guarda acum. (cadenas)
	PUSH, POP	De pila (meter y sacar)
	PUSHF, POPF	Id. (registro de estado)
	XCHG	Intercambio
	XLAT	De acceso a tabla
	SAHF, LAHF	De registro de estado
	LEA, LDS, LES	De direcciones
	IN, OUT	De entrada y salida
LÓGICAS	AND	Producto lógico
	OR	Suma lógica
	XOR	Suma lógica exclusiva
	NOT	Negación (inversión)
	TEST	Comprobación
ARITMÉTICAS	ADD, ADC	Suma (sin y con acarreo)
	SUB, SBB	Resta (sin y con préstamo)
	NEG	Negativación
	INC, DEC	Incremento y decremento
	MUL, IMUL	Multiplicación (s/c signo)
	DIV, IDIV	División (sin/con signo)
	DAA, DAS	Ajuste BCD (empaqueta.)
	AAA, AAS, AAM, AAD	Ídem (desempaquetado)
	CBW, CWD	Extensión de signo
COMPARACIÓN	CMP	Comparación
	SCASB, SCASW	Acumulador con cadena
	CMPSB, CMPSW	Dos cadenas
ROTACIÓN Y DESPLAZAMIENTO	ROR, ROL, RCR, RCL	Rotaciones varias
	SHR, SHL, SAR	Desplazamientos varios
CONTROL DE FLUJO	JMP	Salto incondicional
	Jcondición	Saltos condicionados
	CALL	Llamada a subrutina
	RET, RETF	Retorno de subrutina
	INT, IRET	Interrupción, retorno
	LOOP, LOOPE, LOOPNE	Gestión de bucles
	REP, REPE, REPNE	Ídem (cadenas)
CONTROL DEL SISTEMA	CLC, STC, CMC	Bandera de acarreo
	CLI, STI	Bandera de interrupción
	CLD, STD	Bandera de dirección
	NOP	No operación
	ESC	Escape

2.3.4 Modos de direccionamiento y formatos

DIRECCIONAMIENTO DE MEMORIA

- Los registros del 8086 son de 16 bits, por lo tanto el número de direcciones posibles a direccionar con 1 solo registro es:

$2^{16} = 65536_{10} = 10000_{16}$

Lo cual representa un total de 64 Kbytes y los valores de direcciones se encuentran en el rango de 0 a FFFF.

- Para superar este límite se utilizan 2 registros para direccionar memoria:

Uno de SEGMENTO y otro de DESPLAZAMIENTO (offset) dentro del segmento.

La notación utilizada para una dirección segmentada es:

- SEGMENTO:DESPLAZAMIENTO
  La relación entre la dirección de memoria real y la dirección segmentada es: DIR = SEGMENTO * 16 + DESPLAZAMIENTO
  Al multiplicar por 16 se obtienen 4 bits más con lo que ahora se tiene:

$$2^{20} = 1048576_{10} = 100000_{16}$$

Con lo cual tenemos un total de 1024Kb = 1Mb de memoria direccionable.

Los valores para las direcciones reales se encuentran en el rango

0 a FFFFFh.

MODOS DE DIRECCIONAMIENTO

- Se entiende por modos de direccionamiento a las diferentes formas que pueden tomar los parámetros de las instrucciones del procesador.

REGISTRO

Un parámetro que direcciona a un registro está utilizando el modo de direccionamiento REGISTRO.

Ej: MOV Ax,Bx
En este ejemplo los dos parámetros direccionan un registro.

VALOR o INMEDIATO

- El modo de direccionamiento INMEDIATO es utilizado cuando se hace referencia a un valor constante. Este se codifica junto con la instrucción. Es decir dicho parámetro representa a su valor y no a una dirección de memoria o un registro que lo contiene.

Ej: MOV Ax, 50
En este ejemplo el número 50 es un parámetro inmediato.

DIRECTO

- Se utiliza el modo directo cuando se referencia a una dirección de memoria y la misma esta codificada junto con la instrucción.
- Ej: MOV Al, [127]

En este ejemplo el desplazamiento de la dirección de memoria se codifica junto con la instrucción y el segmento se asume a DS.

Si MEMORIA es un array de bytes que representa a la memoria la instrucción anterior se puede poner como:

Al := MEMORIA[DS:127]

INDIRECTO

- Se utiliza el modo directo cuando se referencia a una dirección de memoria a través de uno o varios registros
- Ej: MOV Al,[Bx]
- Aquí el offset de la dirección de memoria está contenido en el registro Bx y al igual que el caso anterior como no se especifica el segmento se asume DS.
- Si MEMORIA es un array de bytes que representa a la memoria la instrucción anterior se puede poner como:

Al := MEMORIA[DS:Bx]

INDIRECTO

- Ejemplos:

Mov Ax, [Bp + 3]

Add [Bx + Si], 4
Sub Es:[Bx + Di + 5],Dx

IDENTIFICA DIVERSOS TIPOS DE DIRECCIONAMIENTO

- 1 .model small
  2 .stack 128
  3 .data
  4 mensaje1 db “Escribe la primera cadena: ”, 0ah, 0dh, “$”
  5 mensaje2 db “Escribe la segunda cadena: ”, $
  5 .code
  6 .startup
  7   mov ax, 05 (inmediato)
  8   mov dx,ax (Registro)
  9 mov dx,offset mensaje1 (Indirecto)
  10 mov ah,09h (valor o inmediato)
  11 int 21h
  12 lea dx,mensaje2 (directo)
  13 mov ah,09h (valor o inmediato)
  14 int 21h
  15 mov ah,01h (valor o inmediato)
  16 int 21h
  17 mov dl, al (Registro)
  18 mov ah,02h (valor o inmediato)
  19 int 21h

  20 MOV AX, [SI+400h] (Indirecto)
  21 mov ah,04Ch (valor o inmediato)
  22 int 21h
  23 .exit
  24 end