Grado en
Ingeniería Informática

Tema 2:
Instrucciones.
Lenguaje del computador

Unidad 1:
Lenguaje máquina

Rafael Casado González
Rosa María García Muñoz
María Teresa López Bonal
Universidad de Castilla–La Mancha

Arquitectura RISC-V
Arquitectura frente a implementación

- Arquitectura

repertorio de instrucciones	cantidad y variedad de registros	modelo de excepciones
manejo de memoria virtual	mapa de direcciones físicas	etc.

- Implementación
  - Forma en que los procesadores específicos aplican la arquitectura

Arquitectura RISC-V
Arquitectura frente a implementación

- La arquitectura RISC-V está desacoplada de las posibles implementaciones hardware específicas
- Los fabricantes tienen libertad para crear sus propios diseños
  - Pueden implementar un conjunto de instrucciones base
  - y luego agregar extensiones específicas
    - Operaciones de punto flotante
    - SIMD (Single Instruction, Multiple Data)
    - Criptografía
    - …
- Esta modularidad permite personalizar sus implementaciones para satisfacer requisitos específicos

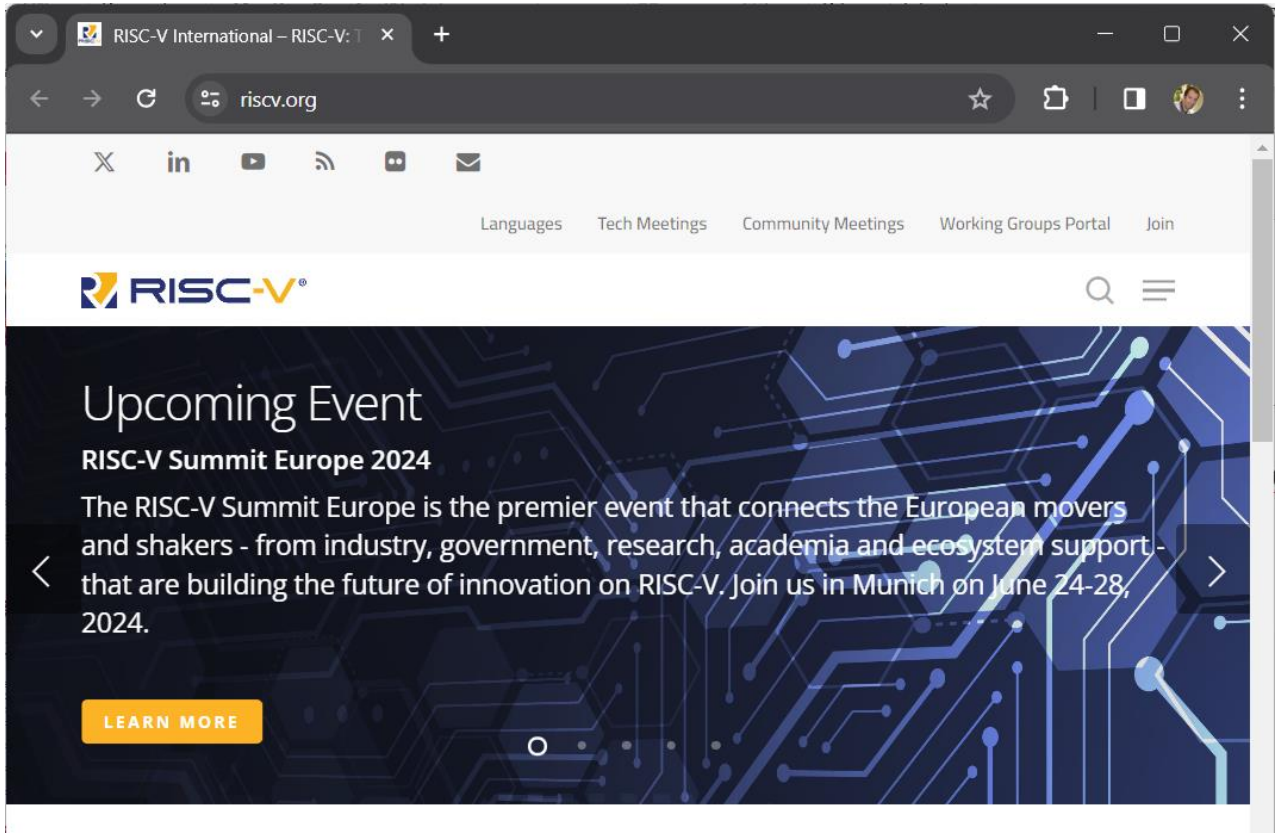
Arquitectura RISC-V
Arquitectura de código abierto

- Promueve la innovación y la colaboración en la industria
  - cualquiera puede acceder a las especificaciones de la arquitectura y contribuir con mejoras o extensiones
  - Esto contrasta con otras arquitecturas propietarias, donde el acceso a las especificaciones puede estar restringido

- Promueve la flexibilidad y la personalización
  - lo que la hace atractiva para una amplia gama de aplicaciones y entornos de desarrollo.

Arquitectura RISC-V
ISA (instruction set architecture)

- Arquitectura del juego de instrucciones:
  - Abierta
  - No propietaria
  - En evolución
- Desarrollada en la Universidad de Berkeley en 2010
- Actualmente coordinada por el consorcio **RISC-V International**
  - riscv.org



Arquitectura RISC-V
RISC (Reduced Instruction Set Computer)

- Repertorio reducido de instrucciones simples
- Gran número de registros de propósito general
- Conjunto reducido de modos de direccionamiento
  - Sólo las instrucciones de carga y almacenamiento acceden a memoria
  - El resto trabajan con datos almacenados en registros

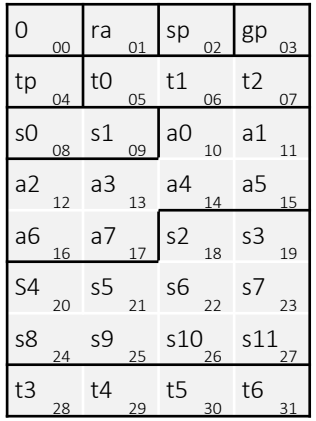
Arquitectura RISC-V
Repertorio base RV32I

- Asume instrucciones de tamaño fijo de 32 bits
- Conjunto mínimo de instrucciones que el procesador debe soportar, incluyendo
  - operaciones en aritmética entera (sumas, restas, multiplicación y divisiones)
  - Operaciones lógicas (AND, OR, XOR)
  - Transferencia de datos (cargar y almacenar)
  - Operaciones de salto condicional e incondicional
- Extensión RVM
  - Aporta multiplicación y división sobre enteros

RISC-V

Registros generales

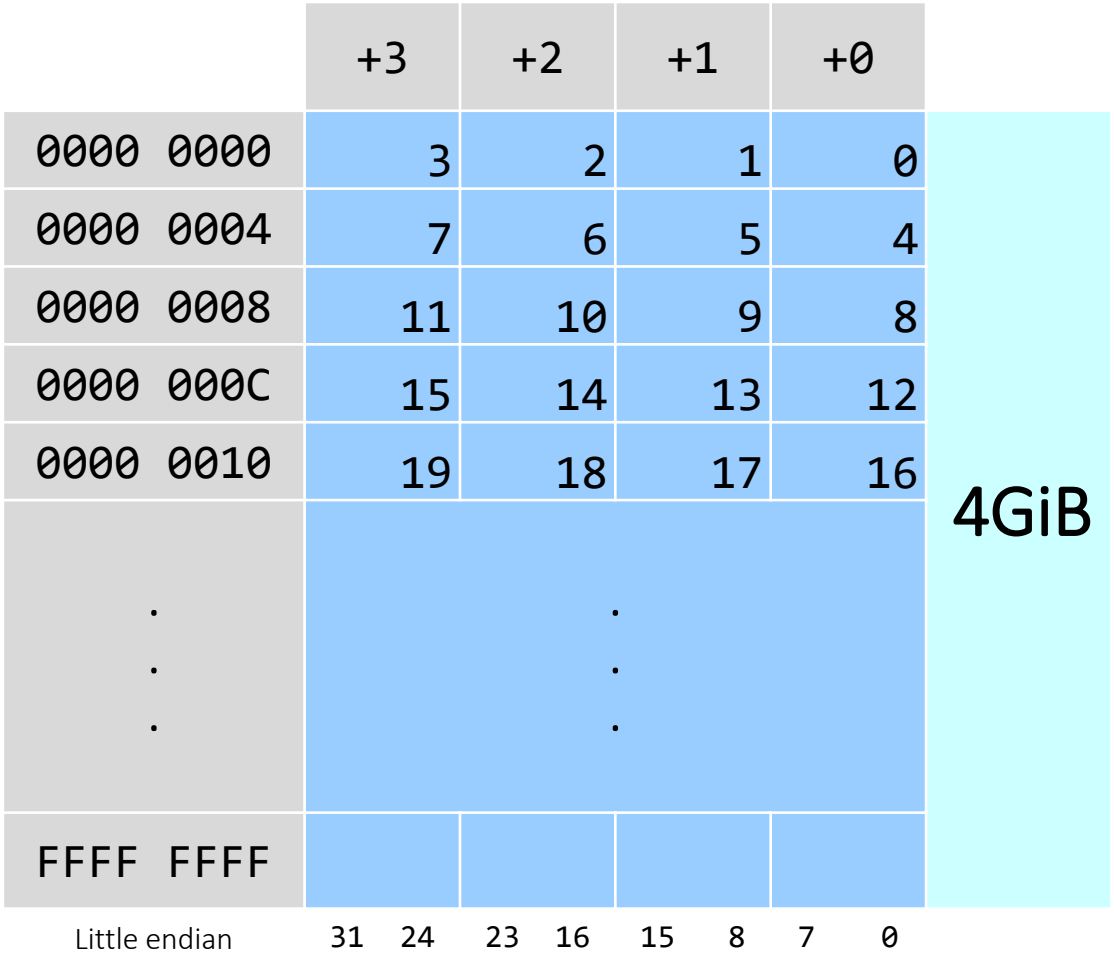
- 32 registros de 32 bits



Número	Alias	Descripción
x0	zero	cableado a 0x00000000
x1	ra	dirección de retorno de subrutina
x2	sp	puntero de pila stack pointer
x3	gp	Puntero global
x4	tp	Puntero de hebra task pointer
x5-x7	t0-t2	registros temporales
x8	s0/fp	registro preservado / puntero de marco
x9	s1	registro preservado
x10-x11	a0-a1	registros de argumento/retorno de valores
x12-x17	a2-a7	registros de argumento
x18-x27	s2-s11	registro preservado
x28-x31	t3-t6	registros temporales

RISC-V
Distribución de memoria

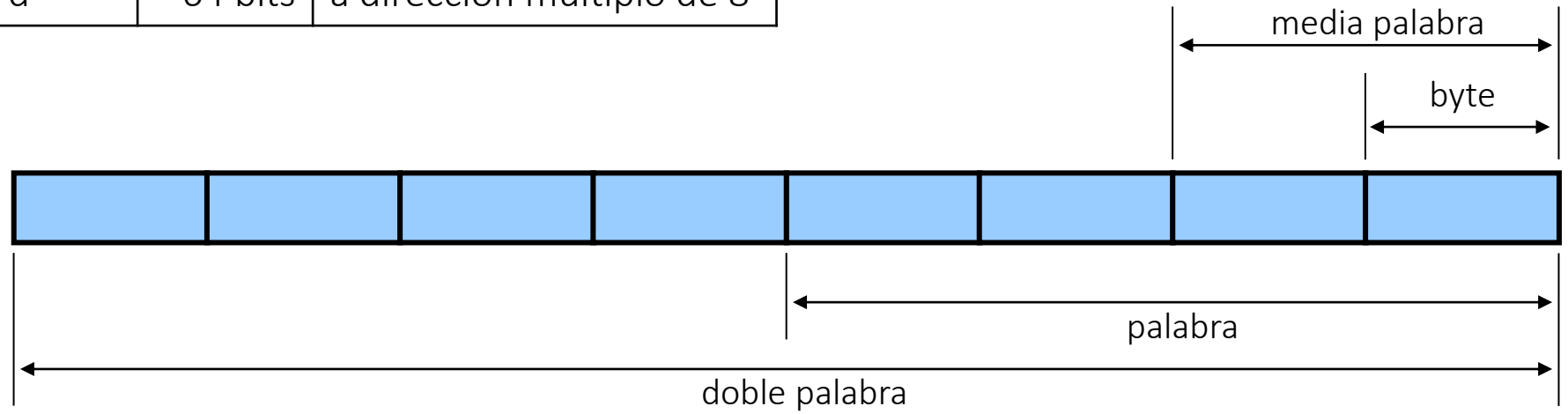
- Bus de direcciones de 32 bits
  - Direccionamiento byte a byte
  - Capacidad de memoria total de 2^{32} bytes = 4GiB
- Bus de datos de 32 bits
  - Palabras de memoria de 32 bits
- Ordenación Little-endian



RISC-V
Distribución de memoria

- Tamaño de datos

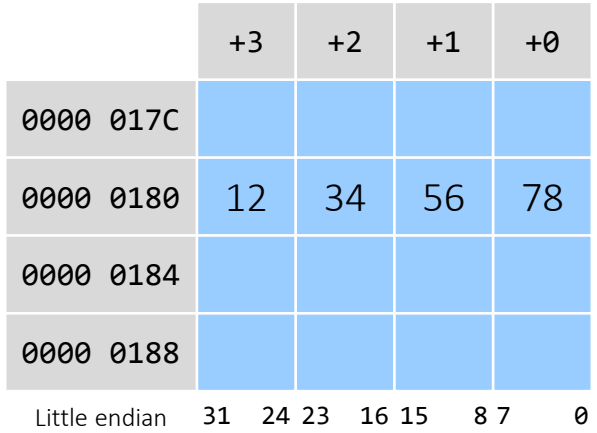
Tipo de dato	Abreviatura	Tamaño	Alineamiento
Byte	b	8 bits	sin alinear
Halfword	h	16 bits	a dirección par
Word	w	32 bits	a dirección múltiplo de 4
Double word	d	64 bits	a dirección múltiplo de 8

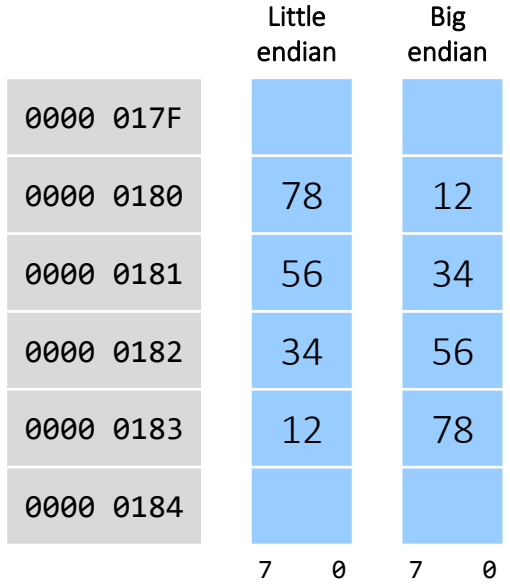


RISC-V
Distribución de memoria

- Ordenación *Little-endian*
  Ejemplo
  - Almacenar el valor $12345678_{(16}$
  - en la posición de memoria $00000180_{(16}$

- La ordenación contraria se denomina *Big-endian*





Formatos de instrucción
Arquitectura del juego de instrucciones

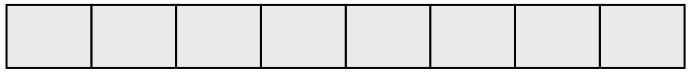
- El ordenador sólo entiende un lenguaje muy restringido y de bajo nivel llamado **lenguaje máquina**
  - Depende del computador
    - Existe una incompatibilidad innata entre distintos computadores
  - Establece sus capacidades básicas
    - Constituye una de las características más importantes de su arquitectura

Formatos de instrucción
Arquitectura del juego de instrucciones

- Las instrucciones máquina
  - Realizan una única y sencilla función
  - Utilizan un número fijo de operandos, con una representación determinada
  - Son autocontenidas
    - Contienen toda la información necesaria para su ejecución
  - Son independientes
    - No requieren información de otras instrucciones para ejecutarse
    - Su interpretación no depende de su posición en el programa o en la memoria

Formatos de instrucción
Arquitectura del juego de instrucciones

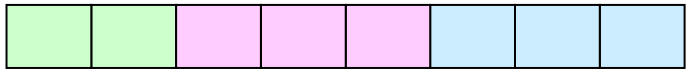
- El formato de una instrucción
  - Especifica el significado de cada uno de los bits que la constituyen
  - Longitud del formato: número de bits que lo componen



- Información que debe contener
  - Operación a realizar
  - Donde encontrar los operandos y el resultado
  - Modo de representación de operandos y resultado
  - Dirección de la siguiente instrucción a ejecutar

Formatos de instrucción
Campos

- En un formato de instrucción los bits se distribuyen en campos
  - Fragmentos de bits contiguos



- Tipos genéricos de campos
  - Campo de código de operación
    - Indica la operación a realizar
    - Es siempre obligatorio y puede extenderse
  - Campos de operandos
    - Puede no existir o haber varios
    - Contienen los operandos (o los lugares donde encontrarlos)

Formatos de instrucción

- Los computadores tienen muy pocos formatos de instrucción
  - Cuando haya varios, el código de operación distinguirá entre ellos
- Los formatos son sistemáticos
  - Fácil codificación y decodificación
  - El primer campo suele ser el código de operación
  - Los tamaños de los campos que expresan direcciones deben corresponder con los mapas direccionados
  - Los tamaños de los formatos encajan en la palabra de memoria de la máquina

Formatos de instrucción
Direccionamiento implícito

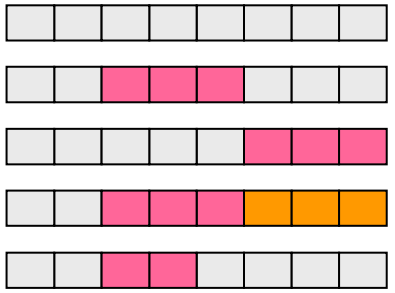
- Las instrucciones cortas son preferibles a las largas
  - Conlleva ahorro de memoria y rapidez de ejecución

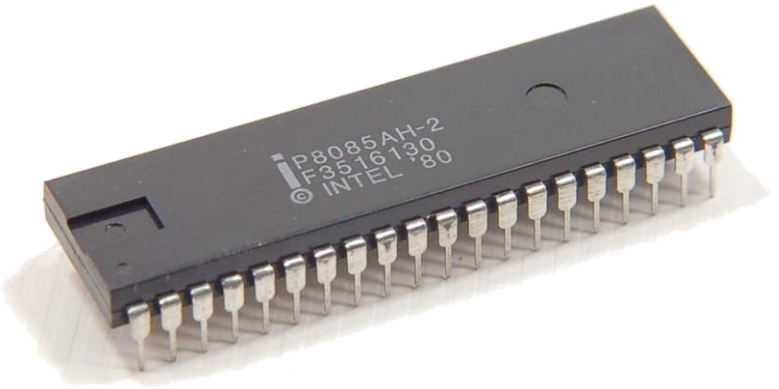
- Para ello se aplica direccionamiento implícito
  - Sólo las instrucciones de bifurcación tienen campo de instrucción siguiente
  - El resultado suele coincidir con el primer operando
  - No se suele especificar la representación de los operandos

Formatos de instrucción

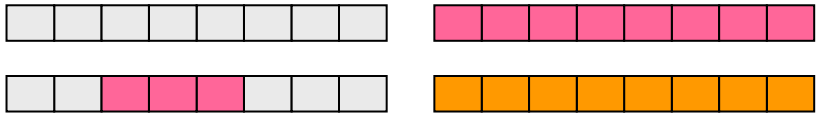
Formatos Intel 8085

- Formatos de 1 byte

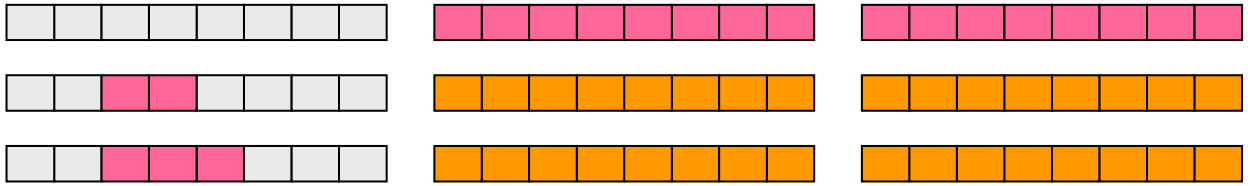


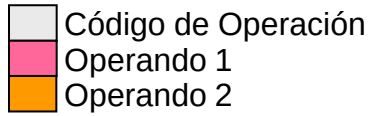


- Formatos de 2 bytes



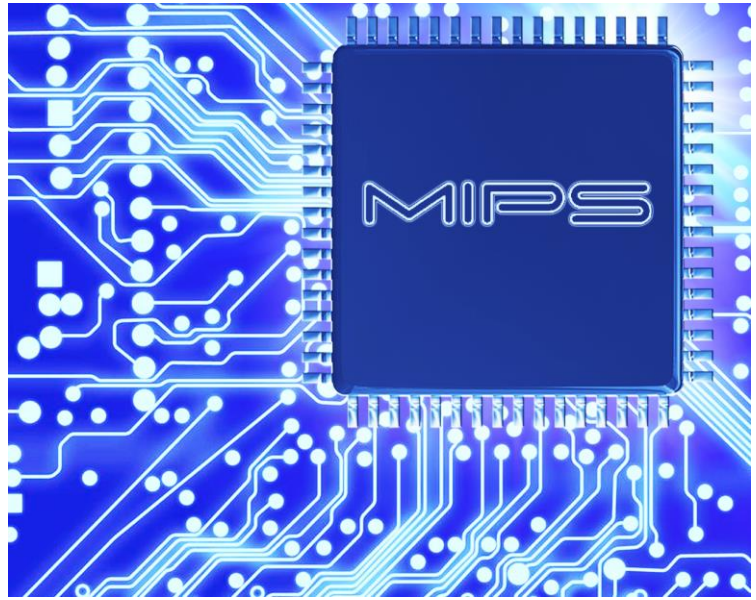
- Formatos de 3 bytes

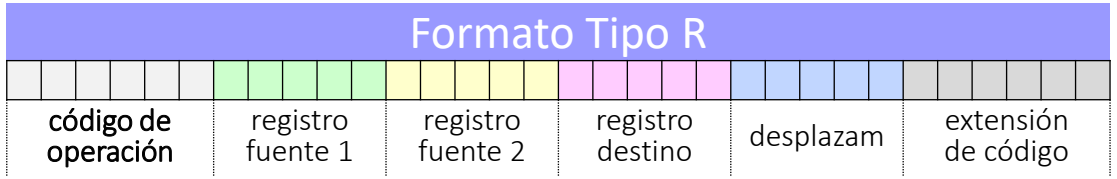


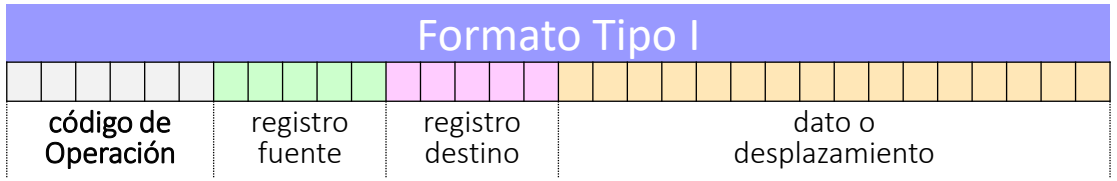


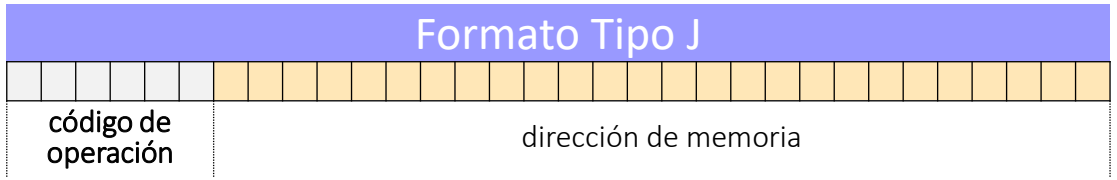
Formatos de instrucción

Formatos MIPS32









Formatos de instrucción

Formatos RISC-V

Formato Tipo R

6	5	4	3	2	1	0	4	3	2	1	0	4	3	2	1	0	2	1	0	4	3	2	1	0	6	5	4	3	2	1	0
función							registro fuente 2					registro fuente 1					función			registro destino						**código de operación**						

Formato Tipo S

11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	2	1	0	4	3	2	1	0	6	5	4	3	2	1	0
dato inmediato							registro fuente 2					registro fuente 1					función			dato inmediato						**código de operación**						

Formato Tipo B

12	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	2	1	0	4	3	2	1	11	6	5	4	3	2	1	0
dato inmediato							registro fuente 2					registro fuente 1					función			dato inmediato						**código de operación**						

Formato Tipo U

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	4	3	2	1	0	6	5	4	3	2	1	0
dato inmediato																				registro destino						**código de operación**						

Formato Tipo J

20	10	9	8	7	6	5	4	3	2	1	11	19	18	17	16	15	14	13	12	4	3	2	1	0	6	5	4	3	2	1	0
dato Inmediato																				registro destino						**código de operación**						

Formato Tipo I

11	10	9	8	7	6	5	4	3	2	1	0	4	3	2	1	0	2	1	0	4	3	2	1	0	6	5	4	3	2	1	0
dato inmediato												registro fuente 1					función			registro destino						**código de operación**						

Modelos de ejecución

- Especifican dónde están almacenados los operandos (y resultado) para las operaciones aritmético-lógicas

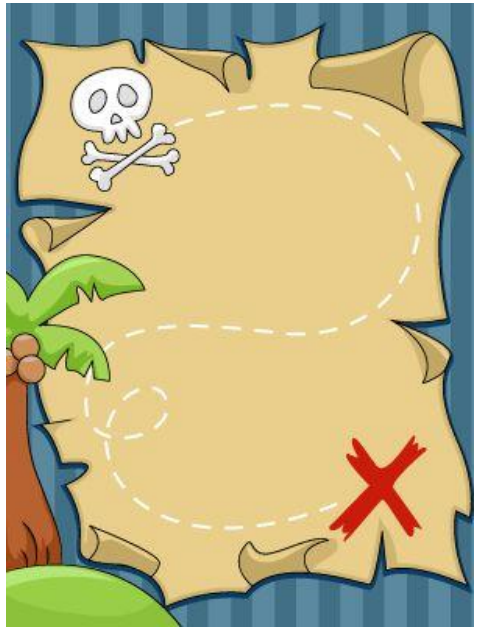
- Están muy ligados al formato de instrucciones
  - Hay máquinas que permiten varios modelos de ejecución
  - Debiendo presentar un formato de instrucción para cada modelo permitido

Modelos de ejecución

- Registro – Registro
  - Sólo se necesita la dirección de los registros
  - Muy rápido
- Registro – Memoria
  - Un operando está en un registro y el otro en memoria
- Memoria – Memoria
  - Operandos y resultado en memoria
- Pila
  - Operandos y resultado en la **pila de máquina**
  - No hace falta dirección (instrucciones PUSH y POP)

Modos de direccionamiento

- Forma de especificar el lugar concreto donde
  - Obtener un operando
  - Almacenar un resultado
  - Realizar un salto



¿Dónde está el operando / resultado / dirección de salto?	¿Qué información contiene la instrucción?
En la propia Instrucción	Ninguna de manera explícita
	El propio operando
En un registro	Ninguna de manera explícita
	El identificador del registro donde está el operando
En la memoria	Ninguna de manera explícita
	La dirección de memoria donde está el operando
	El registro donde está la dirección de memoria donde está el operando
	La dirección de la dirección donde está el operando
En un dispositivo externo	El puerto de E/S correspondiente

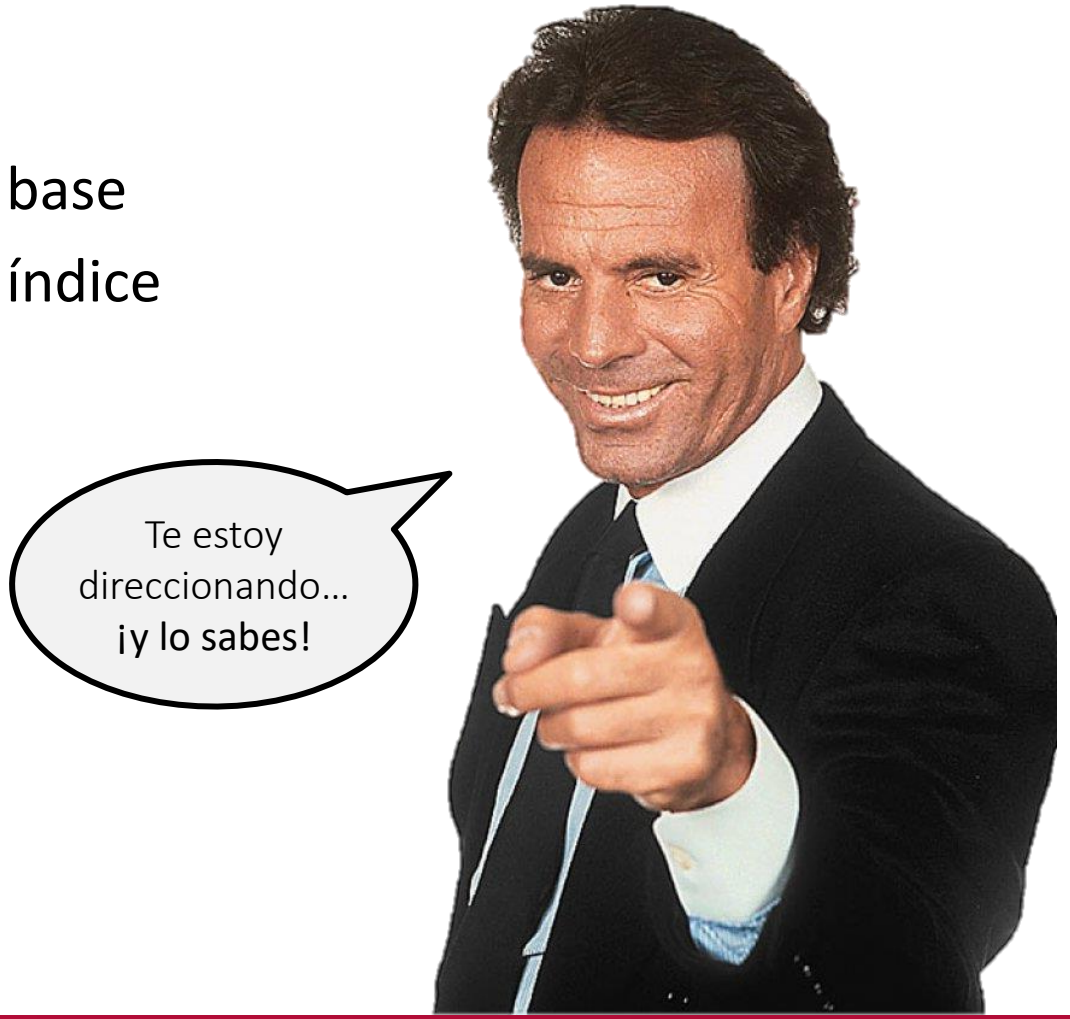
Modos de direccionamiento

- ¿Por qué a veces la instrucción no incluye directamente el operando / resultado / dirección?
  - Para ahorrar espacio en el formato
  - Estructuras de datos dinámicas
    - Listas, pilas, colas,…
    - Vectores, matrices,…
    - Bases de datos
    - Modelos 3D
    - …

Modos de direccionamiento

Tipos

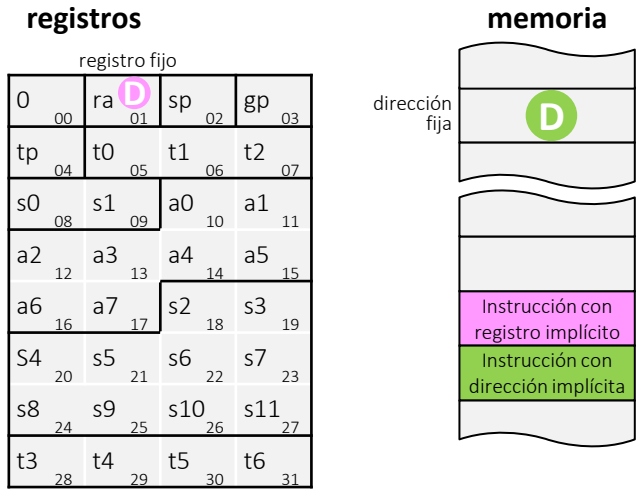
- Implícito
- Inmediato
- Directo (o absoluto)
  - a registro
  - a memoria
- Indirecto (a memoria)
  - por registro
  - por memoria
- Relativo
  - al PC
  - a un registro base
  - a un registro índice
  - a la pila



Modos de direccionamiento

Implícito

- El operando está implícito en el código de operación
  - Normalmente en un registro concreto
  - A veces en direcciones de memoria concretas
- No ocupa espacio en la instrucción
- Restringe la aplicación de la operación a una única opción

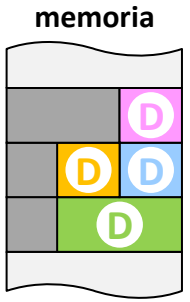


Modos de direccionamiento

Inmediato

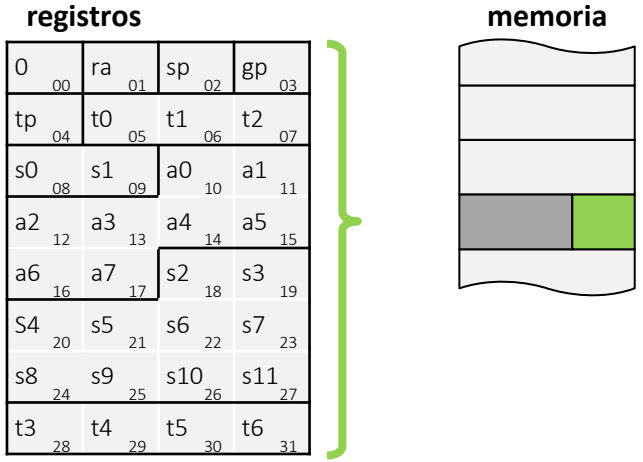
- El operando es un dato contenido en la propia instrucción

- No requiere posteriores referencias a memoria
- El rango del operando está limitado por el tamaño del campo de dirección



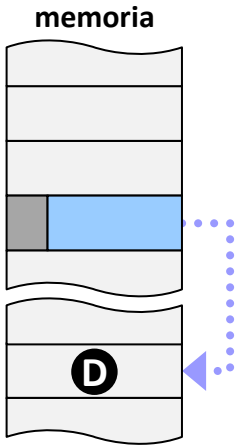
Modos de direccionamiento
Directo (o absoluto) a registro

- El operando queda especificado mediante el registro en el que se haya
  - Un campo del formato identifica al registro



Modos de direccionamiento
Directo (o absoluto) a memoria

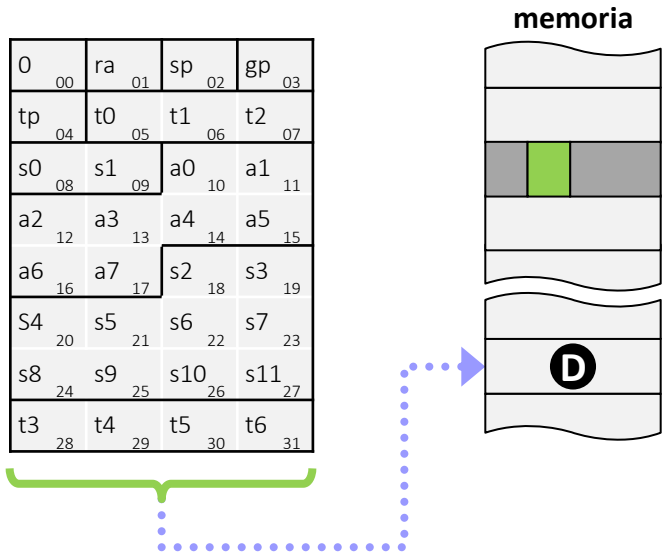
- El operando queda especificado mediante la posición de memoria en la que se haya
  - Un campo del formato identifica la dirección de memoria



- Usado para
  - Manipular datos estáticos
  - Saltos absolutos

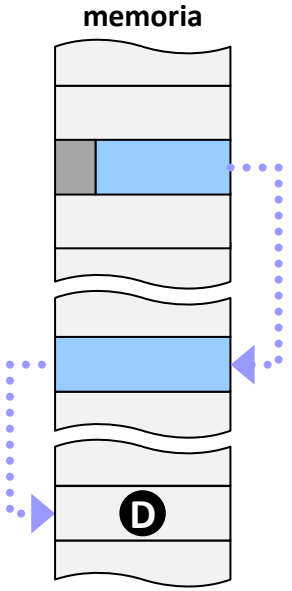
Modos de direccionamiento
Indirecto (a memoria) por registro

- Un registro contiene la dirección de memoria en la que se encuentra el operando
  - Un campo del formato identifica al registro
  - En ocasiones el registro concreto está implícito
    - No requiere bits del formato



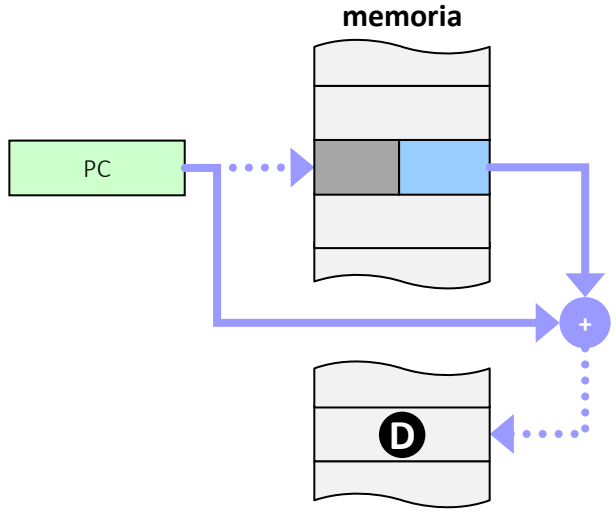
Modos de direccionamiento
Indirecto (a memoria) por memoria

- La instrucción especifica la palabra de memoria que contiene la dirección del operando
  - Se pueden plantear varios niveles de indirección
  - Requiere accesos adicionales a memoria
- Se aplica para acceder a bases de datos mediante una tabla de punteros

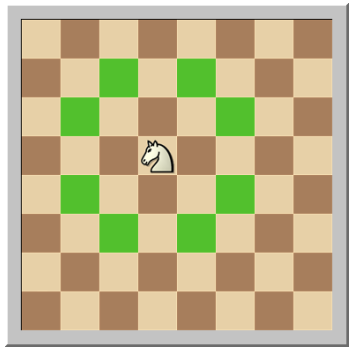


Modos de direccionamiento
Relativo al PC

- El operando se encuentra en memoria, en una posición referenciada con
  - un desplazamiento
    - positivo o negativo
  - respecto a la propia instrucción

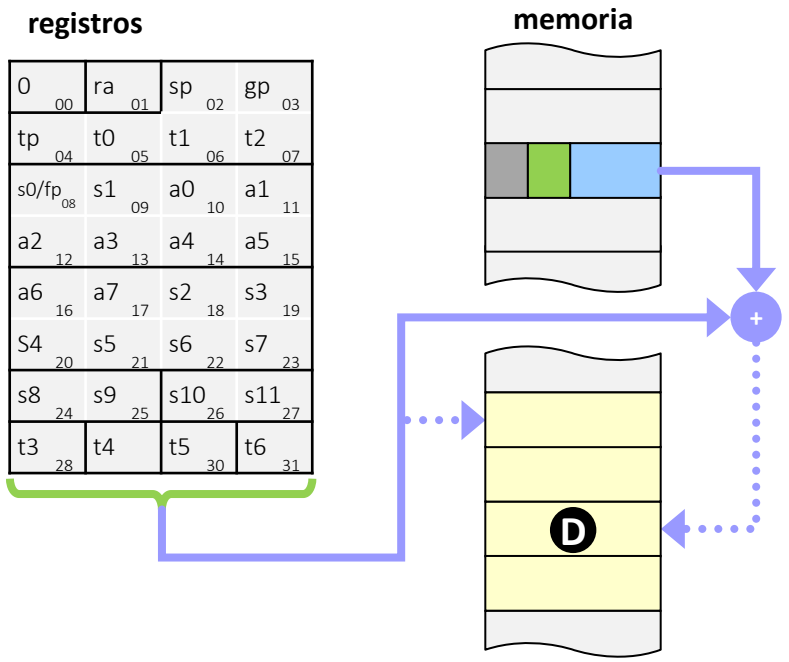


- Suele utilizarse para implementar saltos relativos
  - Que posibilitan el código reubicable



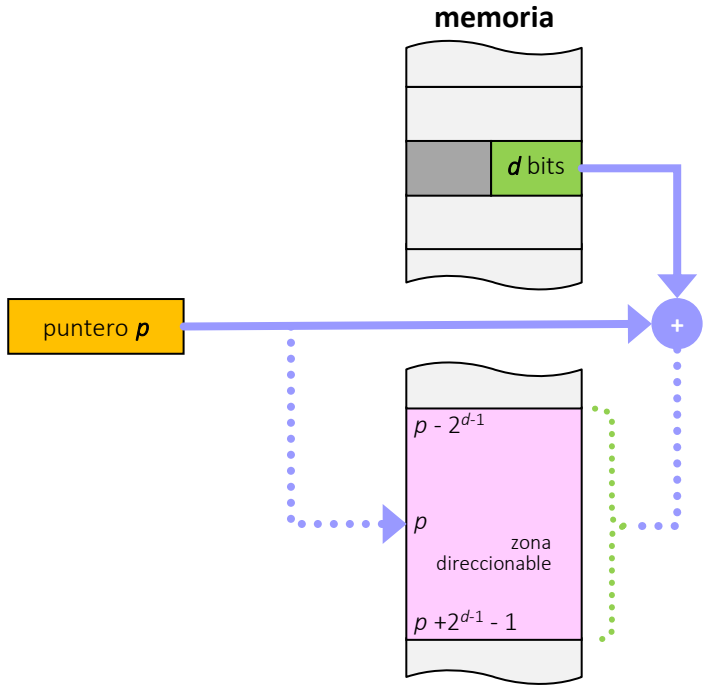
Modos de direccionamiento
Relativo a un registro base

- El operando se encuentra en memoria, en una posición obtenida a partir de
  - una referencia base (extraída de un registro)
  - más un desplazamiento determinado
- Sirve para implementar
  - Acceso complejo a estructuras de datos
  - Protección de memoria



Modos de direccionamiento
Relativo

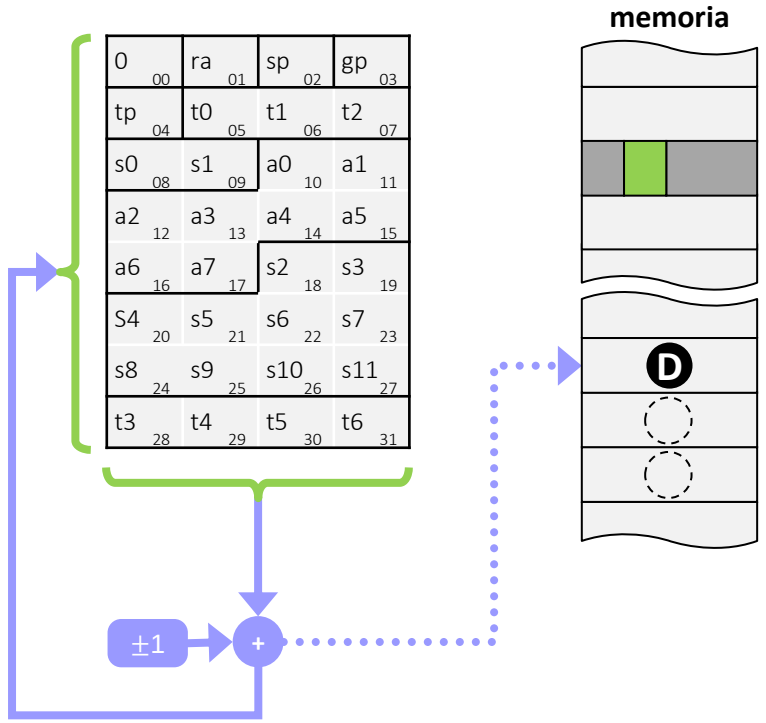
- En los modos de direccionamiento relativos
  - a PC
  - a registro base
- el rango direccionable lo establece
  - la cantidad de bits del campo de desplazamiento
  - En complemento a dos



Modos de direccionamiento
Relativo a un registro índice

- El operando se encuentra en una posición de memoria indicada por un registro que se actualiza automáticamente

	Decremento	Incremento
Pre	`--i`	`++i`
Post	`i--`	`i++`

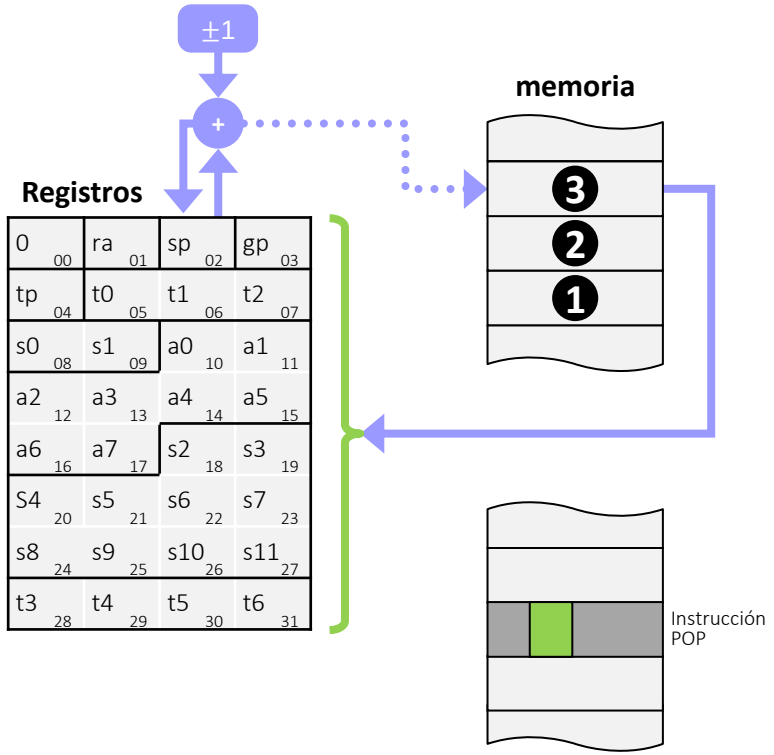


- Permite implementar barridos de memoria

Modos de direccionamiento
Relativo a pila

- El operando se encuentra en la cabecera de la pila de máquina
  - Instrucciones PUSH y POP
  - Acceso mediante el registro SP (Stack Pointer)
    - No requiere bits del formato

- Permite implementar subrutinas anidadas
  - Recursividad



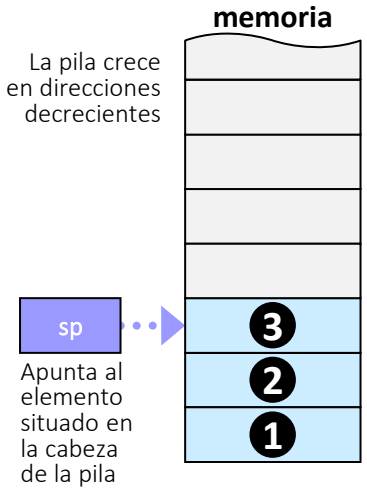
Modos de direccionamiento

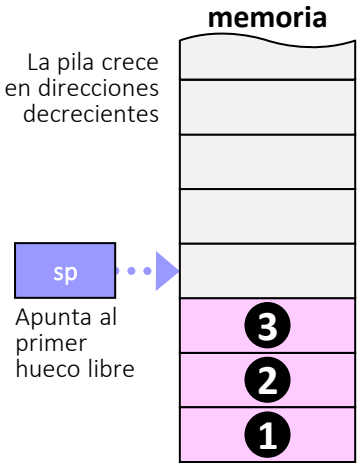
Relativo a pila

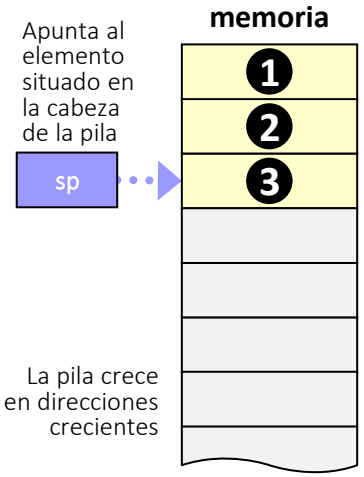
- Instrucciones muy compactas
  - Pues no requiere campo de dirección
- La pila puede crecer de diferentes formas

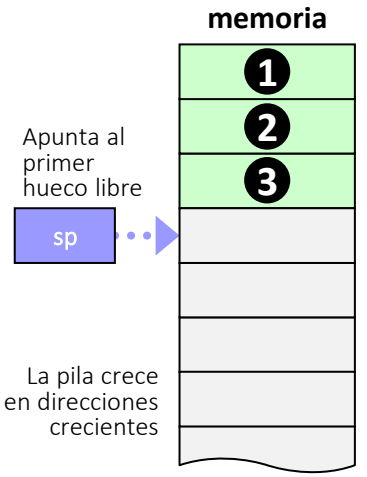
PUSH	Decremento	Incremento
Pre	`--sp`	`++sp`
Post	`sp--`	`sp++`

POP	Decremento	Incremento
Pre	`--sp`	`++sp`
Post	`sp--`	`sp++`



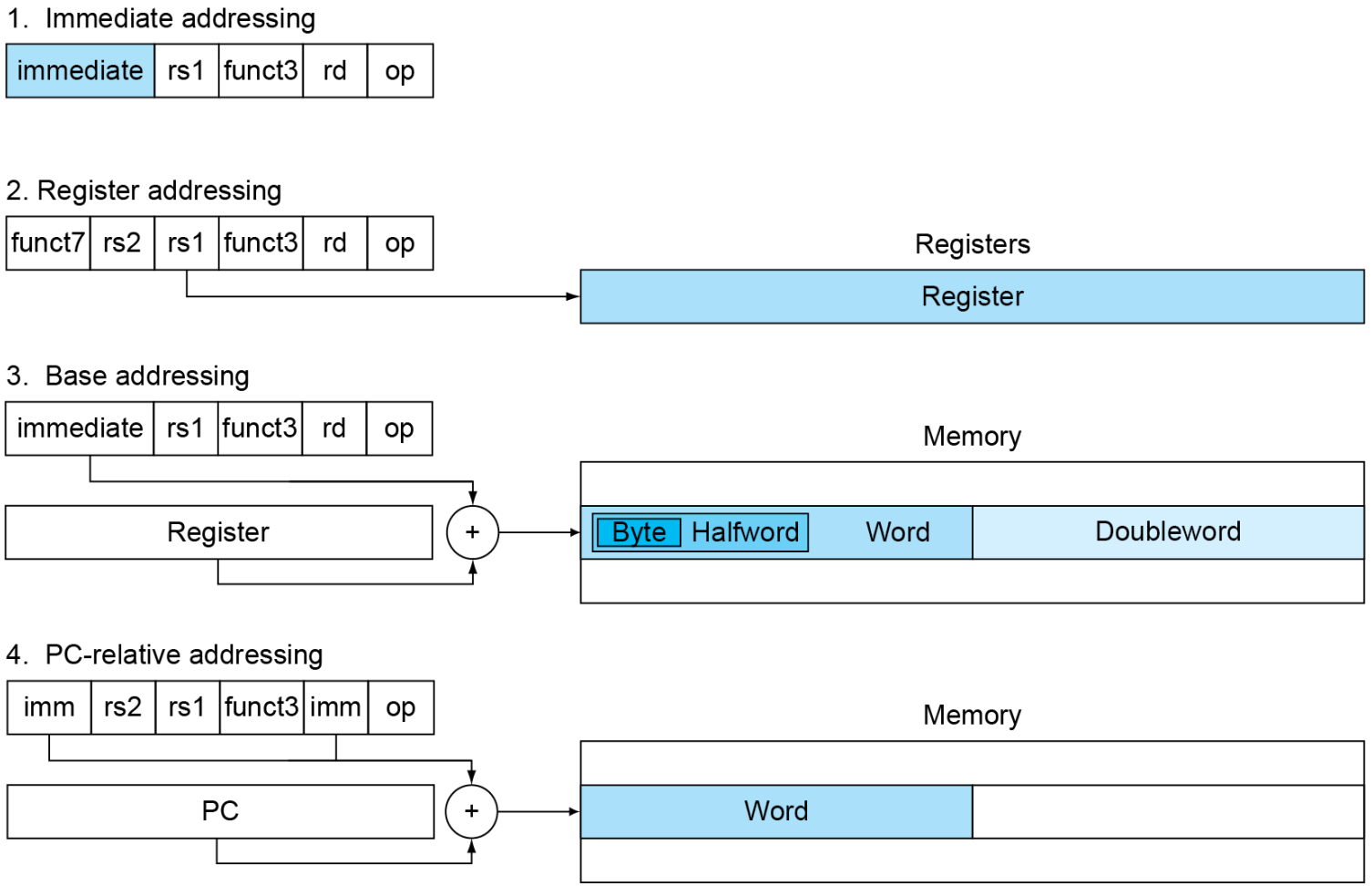






RISC-V Modos de direccionamiento

- Inmediato
- Directo a registro
- Relativo a registro base
- Relativo a contador de programa



RISC-V Modos de direccionamiento
Ejercicios

- Determinar los modos de direccionamiento de estas instrucciones RISC-V

  - `sub  t3,s1,s5`
  - `ori  t6,t6,0xf00`
  - `lw   a1,65(t5)`
  - `bne  zero,t5,0xffc`
  - `jal  ra, repetir`

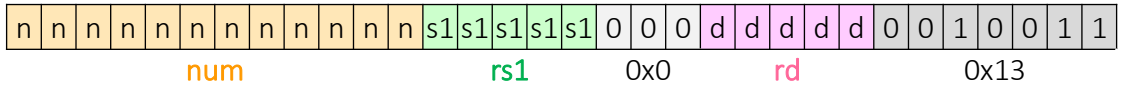
Instrucciones RISC-V

Tipos

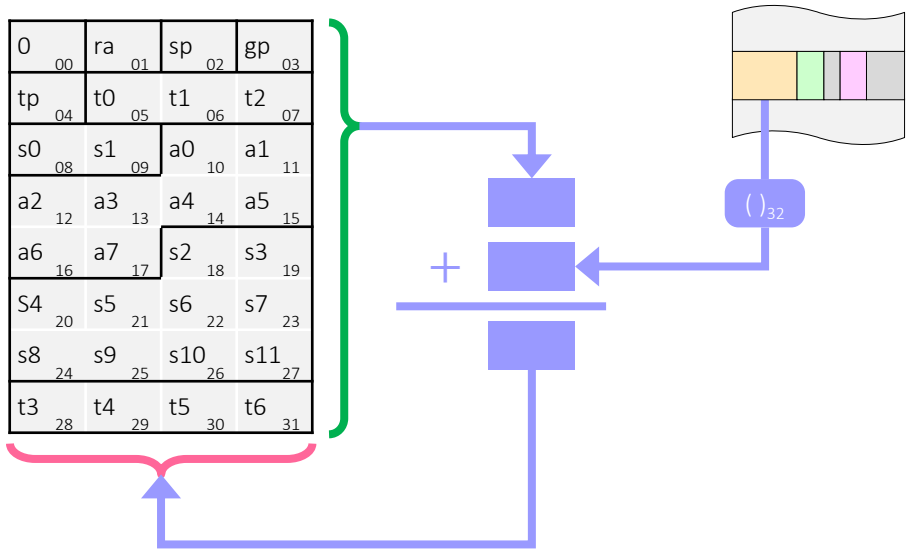
- Operaciones aritméticas
- Operaciones lógicas
- Lectura de memoria
- Escritura en memoria
- Transferencia de datos
- Operación de desplazamiento
- Operaciones de comparación
- Saltos condicionales
- Saltos incondicionales
- Misceláneas (otras)

Instrucciones RISC-V
Operaciones aritméticas

Nemónico	Operación	Formato I
addi rd, rs1, num	rd ← rs1 + $(num)_{32}$	



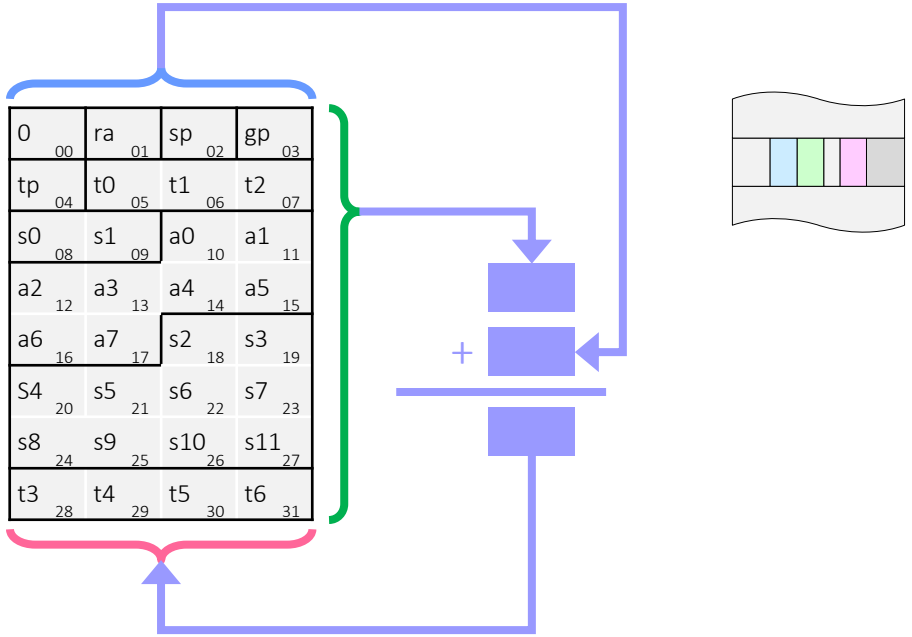
Suma un registro y un dato
en complemento a dos
dejando el resultado en otro registro



Instrucciones RISC-V
Operaciones aritméticas

Nemónico	Operación	Formato R
add rd, rs1, rs2	rd ← rs1 + rs2	0 0 0 0 0 0 0 s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 0 0 0 d d d d d 0 1 1 0 0 1 1
		0x0 rs2 rs1 0x0 rd 0x33

Suma dos registros
dejando el resultado en un tercero

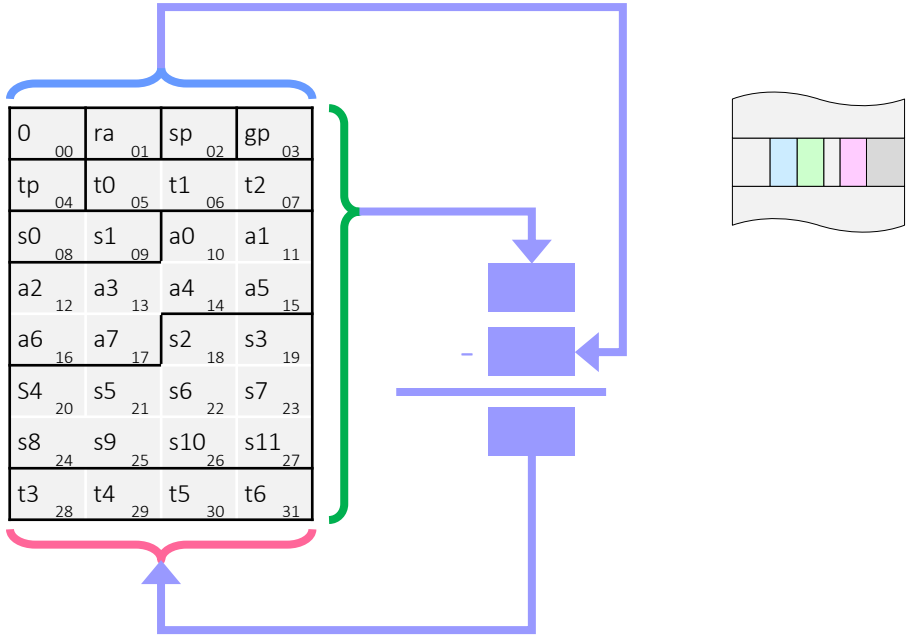


Instrucciones RISC-V
Operaciones aritméticas

Nemónico	Operación	Formato R
sub rd, rs1, rs2	rd ← rs1 - rs2	0 1 0 0 0 0 0 s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 0 0 0 d d d d d 0 1 1 0 0 1 1

0x20 rs2 rs1 0x0 rd 0x33

Resta dos registros
dejando el resultado en un tercero

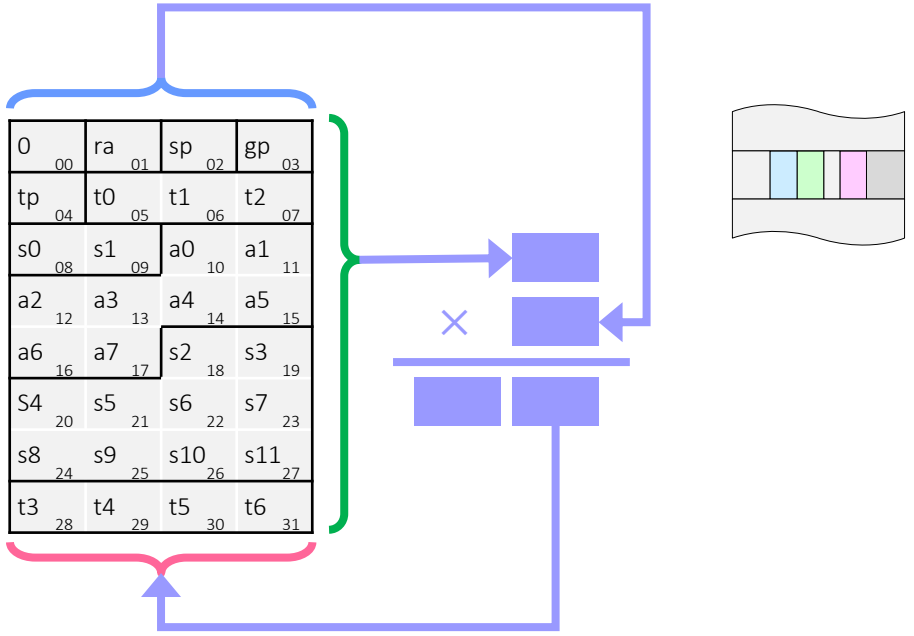


Instrucciones RISC-V

Operaciones aritméticas

Nemónico	Operación	Formato R
mul rd, rs1, rs2	rd ← $(rs1 \times rs2)_{31:0}$	0 0 0 0 0 0 1 s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 0 0 0 d d d d d 0 1 1 0 0 1 1
		0x01 rs2 rs1 0x0 rd 0x33

Multiplica dos registros
dejando la parte baja del resultado
en un tercero

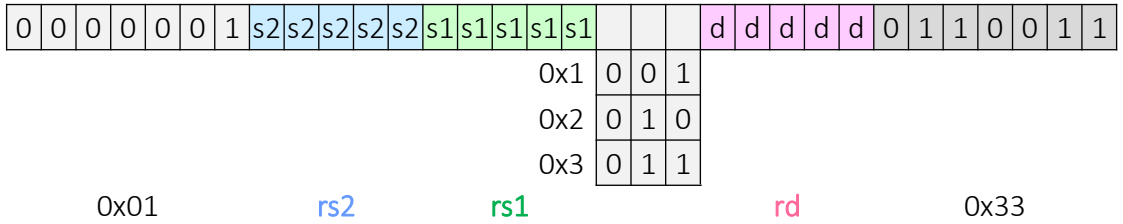


Instrucciones RISC-V

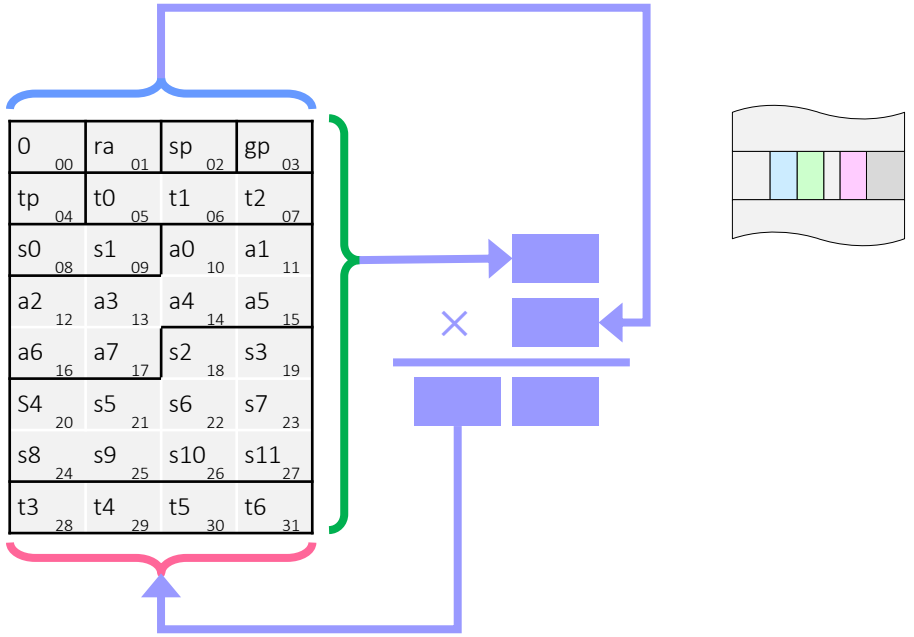
Operaciones aritméticas

Nemónico		Operación
		$rd \leftarrow (rs1 \times rs2)_{63:32}$
mulh	rd, rs1, rs2	signed / signed
mulhsu	rd, rs1, rs2	signed / unsigned
mulhu	rd, rs1, rs2	unsigned / unsigned

Formato R

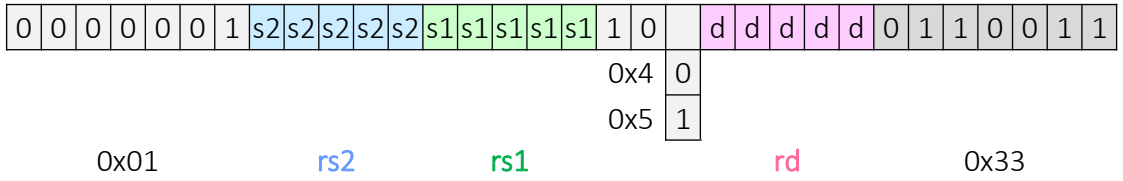


Multiplica dos registros
dejando la parte alta del resultado
en un tercero

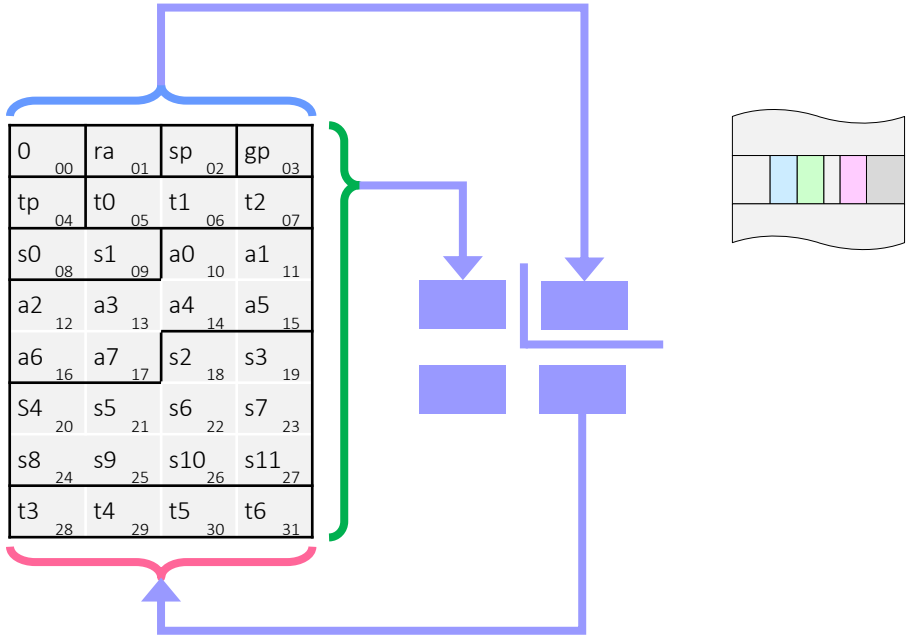


Instrucciones RISC-V
Operaciones aritméticas

Nemónico	Operación	Formato R
	rd ← rs1 / rs2	
div rd, rs1, rs2	en complemento a dos	
divu rd, rs1, rs2	en binario natural	



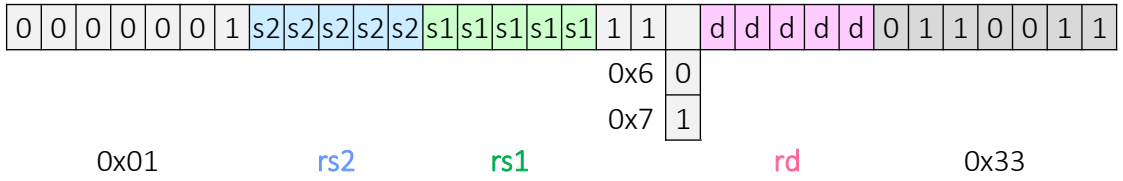
Divide un registro entre otro
dejando el cociente en un tercero



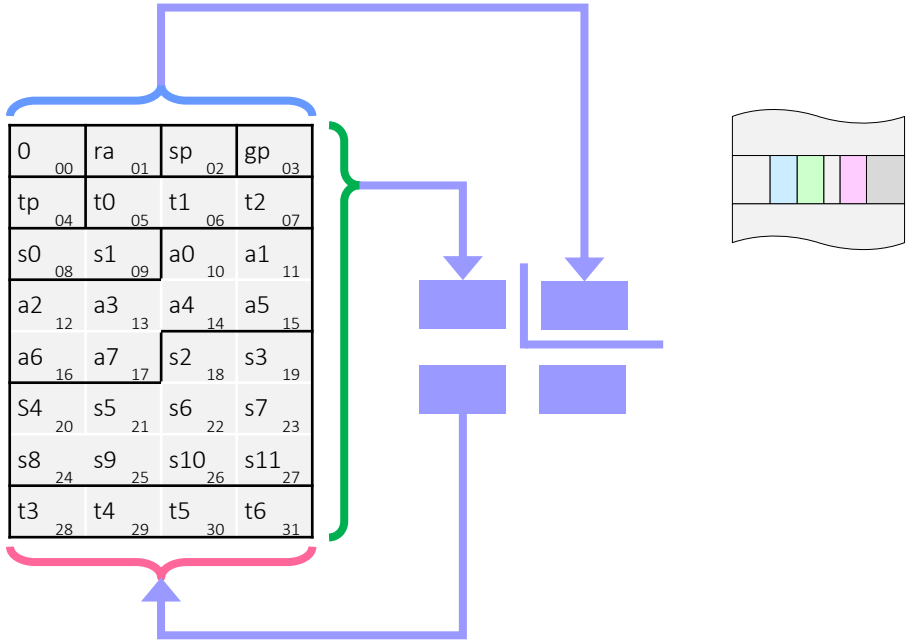
Instrucciones RISC-V
Operaciones aritméticas

Nemónico	Operación
	rd ← rs1 % rs2
rem rd, rs1, rs2	en complemento a dos
remu rd, rs1, rs2	en binario natural

Formato R



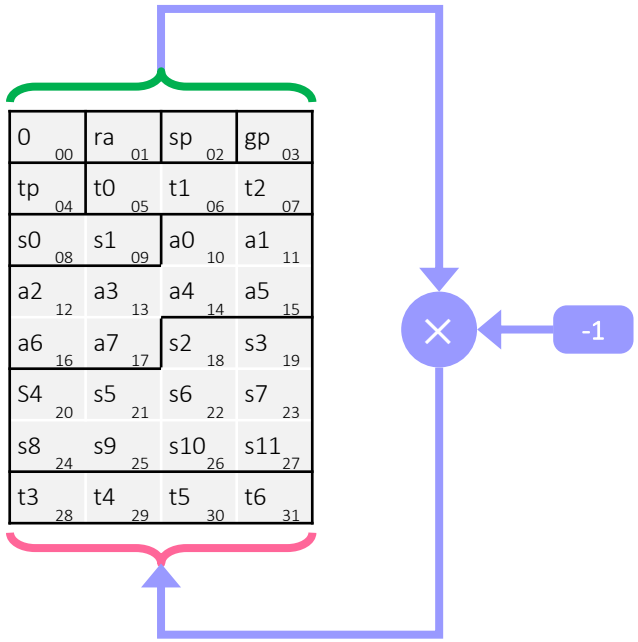
Divide un registro entre otro dejando el resto en un tercero



Instrucciones RISC-V
Operaciones aritméticas

Nemónico	Operación	Pseudo-instrucción
neg rd, rs1	rd ← -rs1	sub rd, zero, rs1

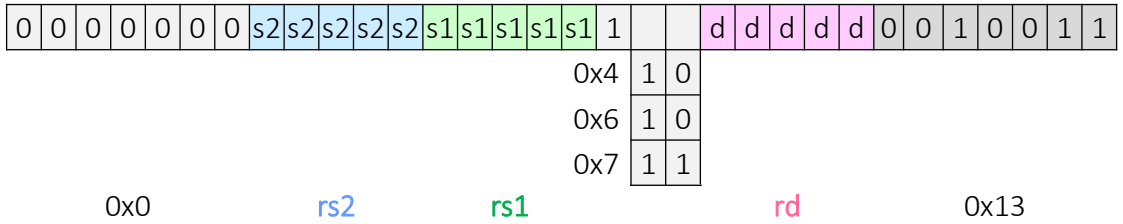
Copia a un registro
el complemento a dos de otro registro



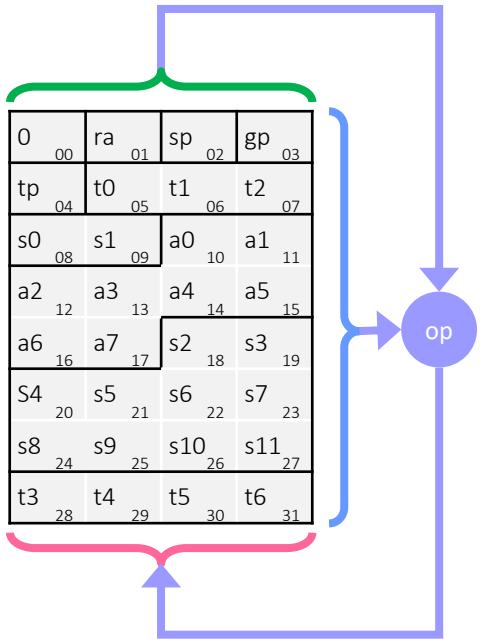
Instrucciones RISC-V
Operaciones lógicas

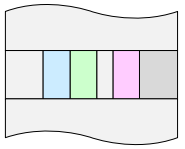
Nemónico	Operación	Formato R

xor	rd, rs1, rs2	rd ← rs1 xor rs2
or	rd, rs1, rs2	rd ← rs1 or rs2
and	rd, rs1, rs2	rd ← rs1 and rs2



Operación lógica
entre dos registros
dejando el resultado en un tercero

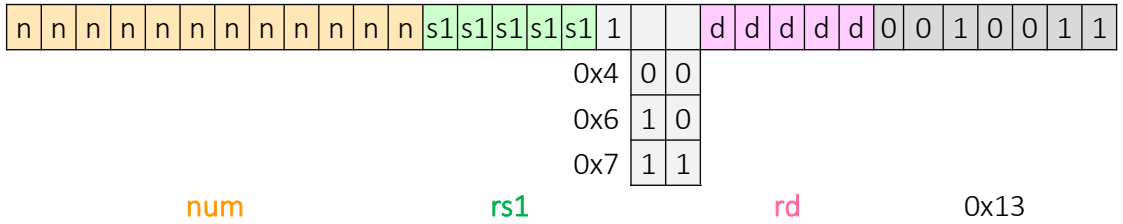




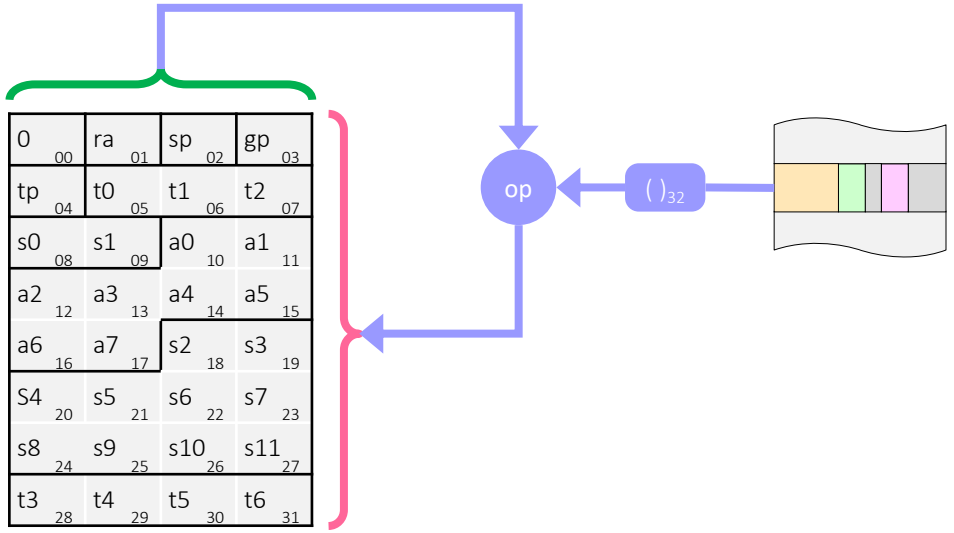
Instrucciones RISC-V
Operaciones lógicas

Nemónico	Operación
xori rd, rs1, num	rd ← rs1 xor $(num)_{32}$
ori rd, rs1, num	rd ← rs1 or $(num)_{32}$
andi rd, rs1, num	rd ← rs1 and $(num)_{32}$

Formato I



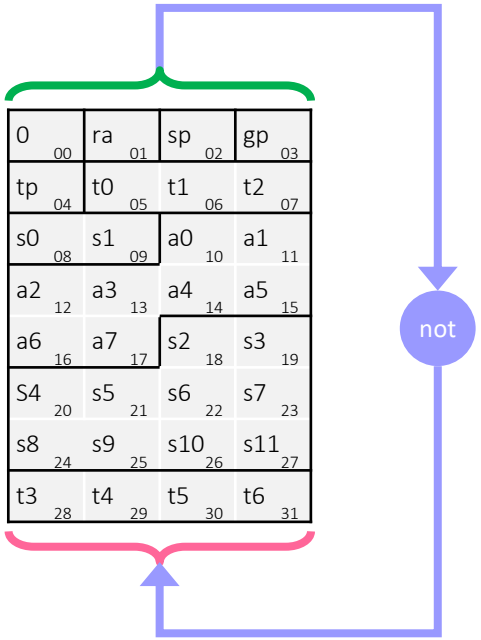
Operación lógica
entre un registro y un dato
dejando el resultado en otro tercero



Instrucciones RISC-V
Operaciones lógicas

Nemónico	Operación	Pseudo-instrucción
not rd, rs1	rd ← not rs1	xori rd, rs1, -1

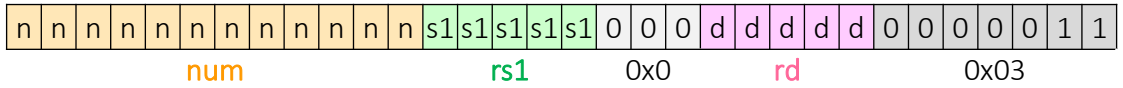
Copia a un registro
el **complemento a uno** de otro registro



Instrucciones RISC-V

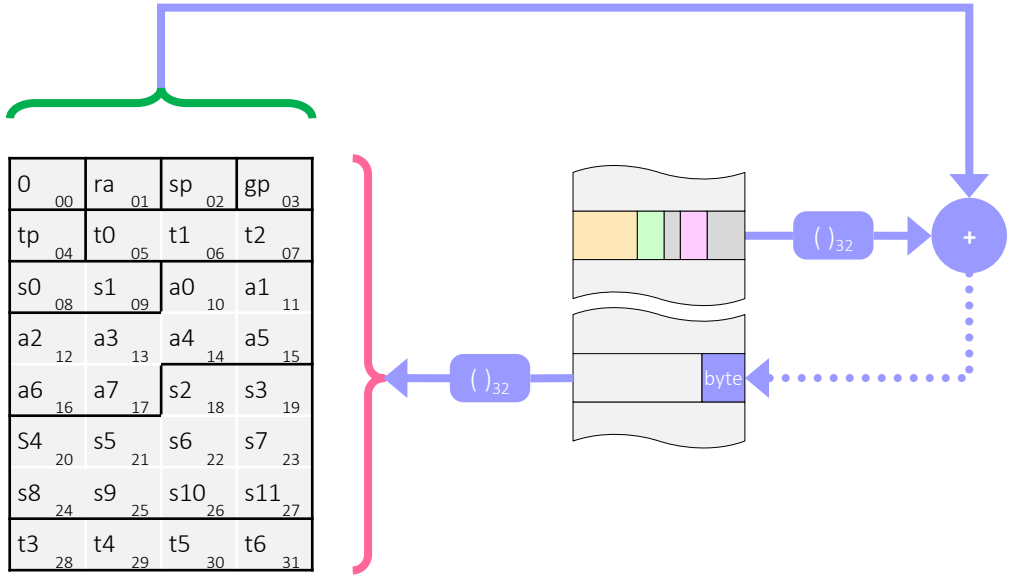
Lectura de memoria

Nemónico	Operación	Formato I
lb rd, num(rs1)	rd ← ([rs1 + (num)$_{32}$]$_8$)$_{32}$	



load byte

Carga en un registro
un octeto de la memoria,
al que se aplica extensión de signo

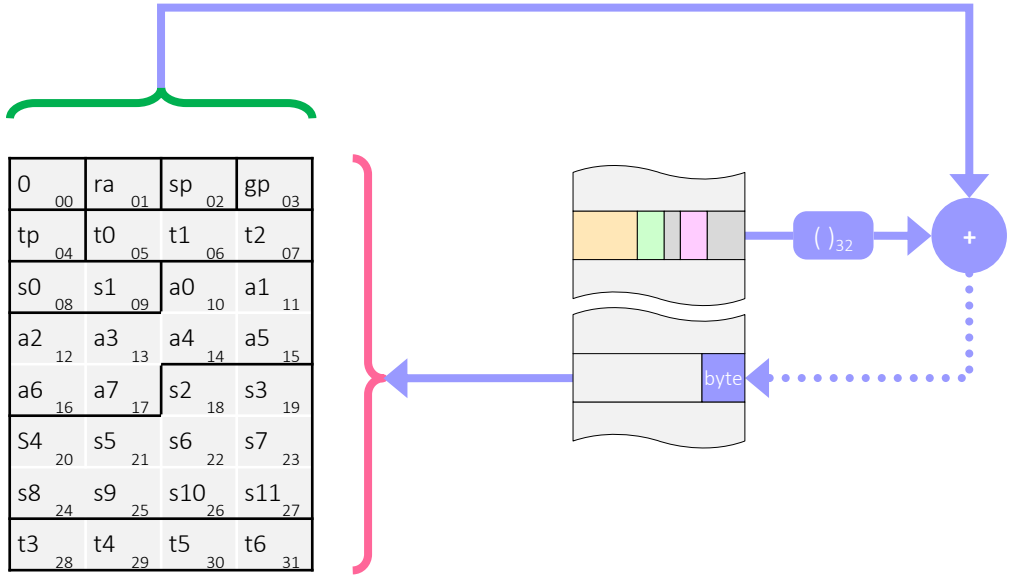


Instrucciones RISC-V
Lectura de memoria

Nemónico	Operación	Formato I
lbu rd, num(rs1)	rd ← [rs1 + (num)$_{32}$]$_8$	n n n n n n n n n n n n s1 s1 s1 s1 s1 1 0 0 d d d d d 0 0 0 0 0 1 1
		num rs1 0x4 rd 0x03

load byte unsigned

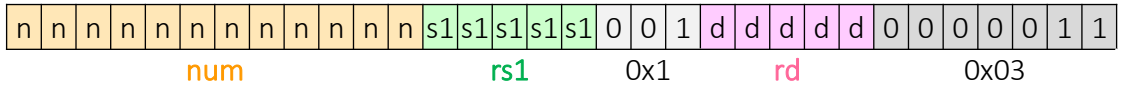
Carga en un registro
un octeto de la memoria,
al que no se aplica extensión de signo



Instrucciones RISC-V

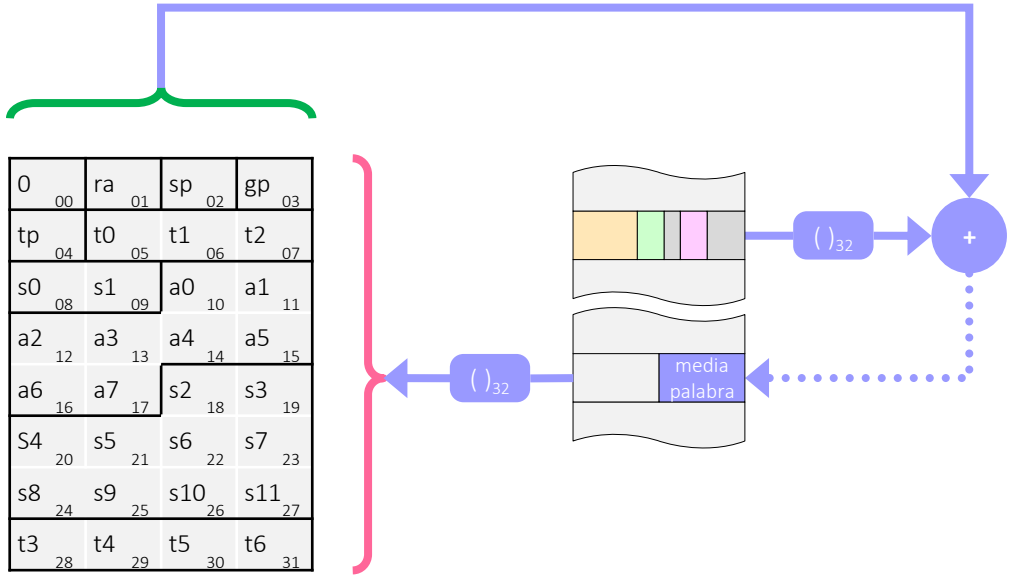
Lectura de memoria

Nemónico	Operación	Formato I
lh rd, num(rs1)	rd ← ([rs1 + (num)$_{32}$]$_{16}$)$_{32}$	



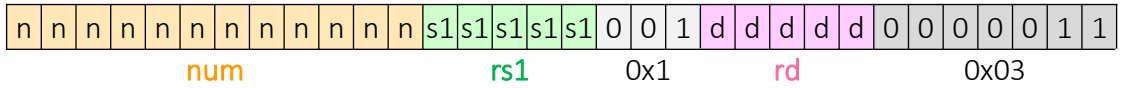
load half

Carga en un registro
media palabra de la memoria,
a la que se aplica extensión de signo



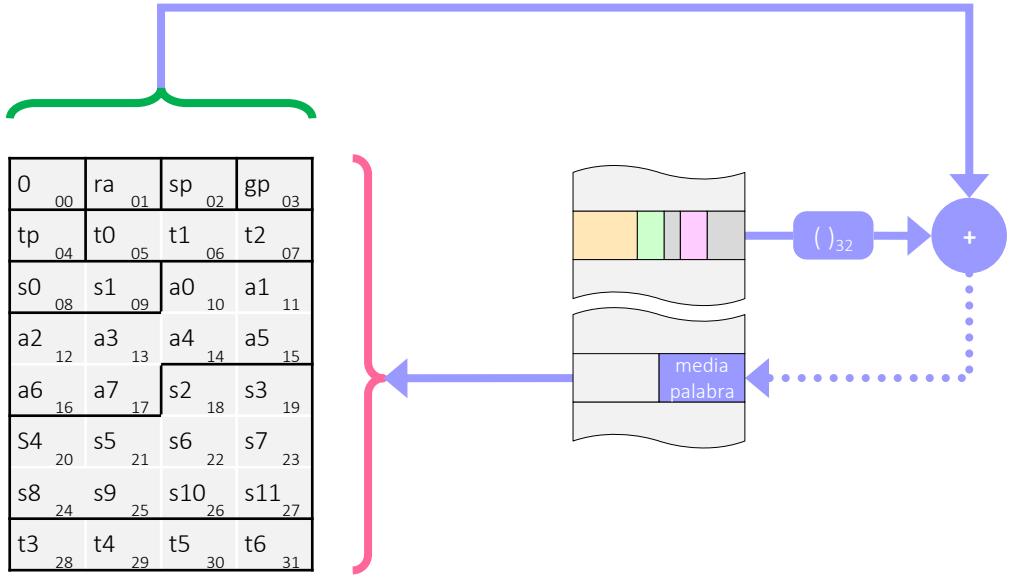
Instrucciones RISC-V
Lectura de memoria

Nemónico	Operación	Formato I
lhu rd, num(rs1)	rd ← [rs1 + (num)$_{32}$]$_{16}$	



load half unsigned

Carga en un registro
media palabra de la memoria,
a la que no se aplica extensión de signo



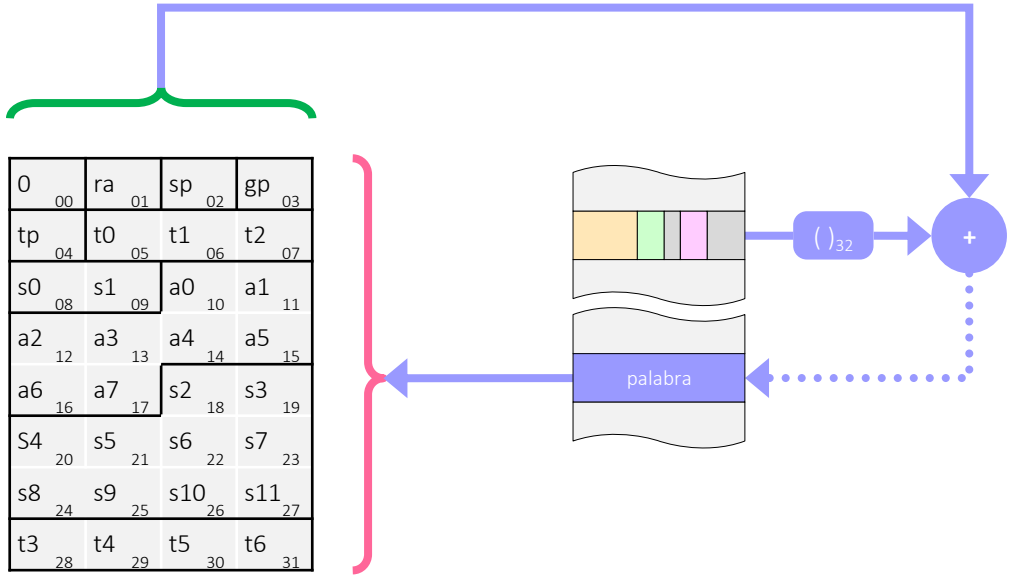
Instrucciones RISC-V

Lectura de memoria

Nemónico	Operación	Formato I
lw rd, num(rs1)	rd ← [rs1 + (num)$_{32}$	n n n n n n n n n n n n s1 s1 s1 s1 s1 0 1 0 d d d d d 0 0 0 0 0 1 1
		num rs1 0x2 rd 0x03

load word

Carga en un registro
Una palabra de la memoria



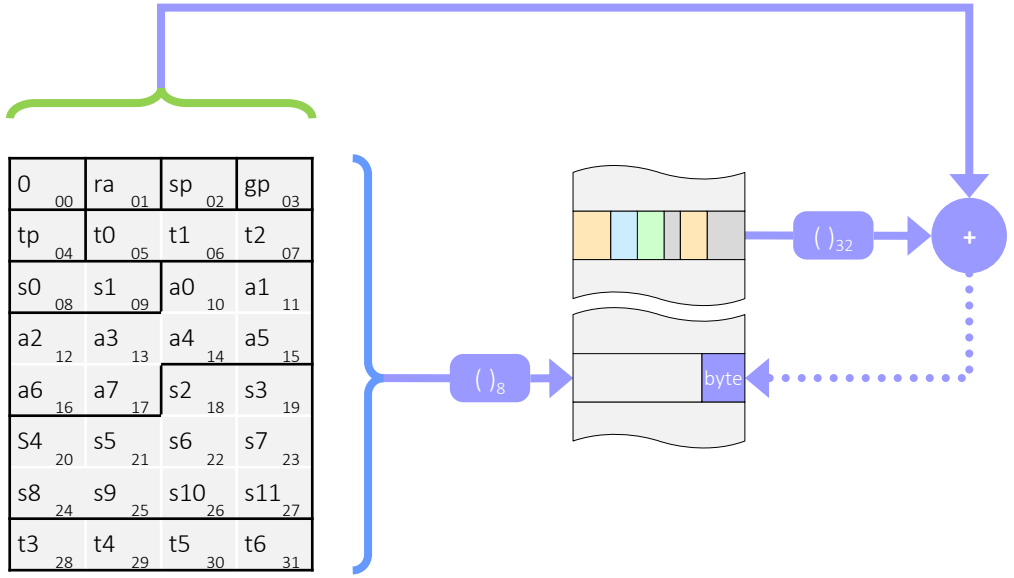
Instrucciones RISC-V

Escritura en memoria

Nemónico	Operación	Formato S
sb rs2, num(rs1)	$[rs1 + (num)_{32}]_8 \leftarrow (rs2)_8$	n n n n n n n s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 0 0 0 n n n n n 0 1 0 0 0 1 1
		num 11:5 rs2 rs1 0x0 num 4:0 0x23

store byte

Almacena un octeto de un registro en la memoria

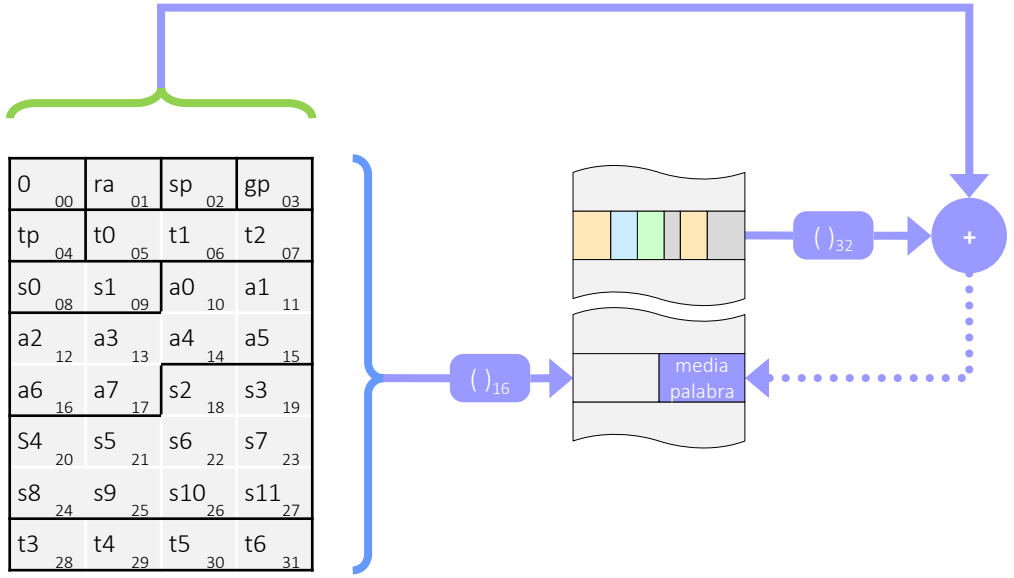


Instrucciones RISC-V
Escritura en memoria

Nemónico	Operación	Formato S
sh rs2, num(rs1)	$[rs1 + (num)_{32}]_{16} \leftarrow (rs2)_{16}$	n n n n n n n s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 0 0 1 n n n n n 0 1 0 0 0 1 1
		num 11:5 rs2 rs1 0x1 num 4:0 0x23

store half

Almacena la media palabra inferior
de un registro en la memoria



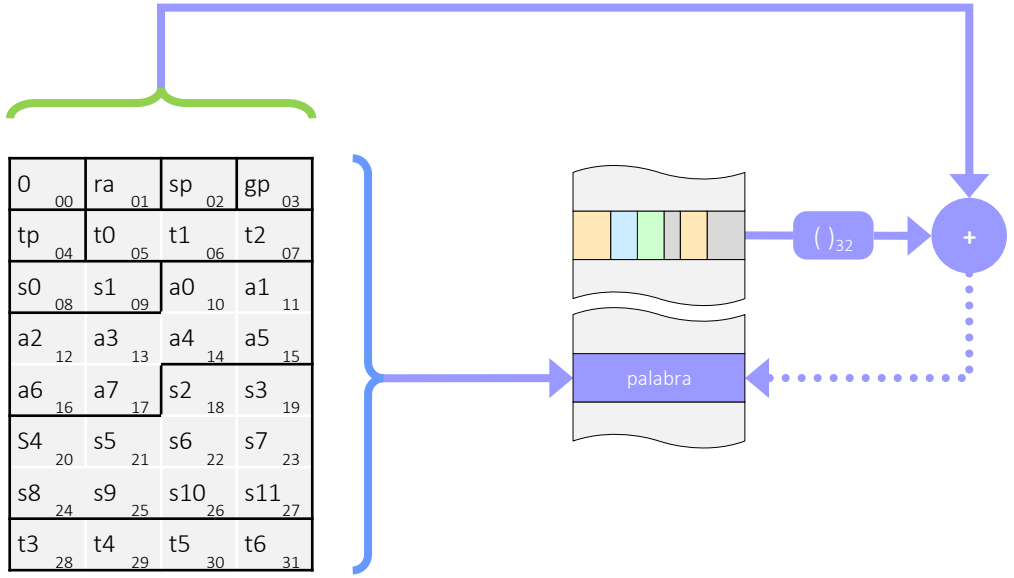
Instrucciones RISC-V

Escritura en memoria

Nemónico	Operación	Formato S
sw rs2, num(rs1)	[rs1 + $(num)_{32}$] ← rs2	n n n n n n n s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 0 1 0 n n n n n 0 1 0 0 0 1 1
		num 11:5 rs2 rs1 0x2 num 4:0 0x23

store word

Almacena un registro en la memoria



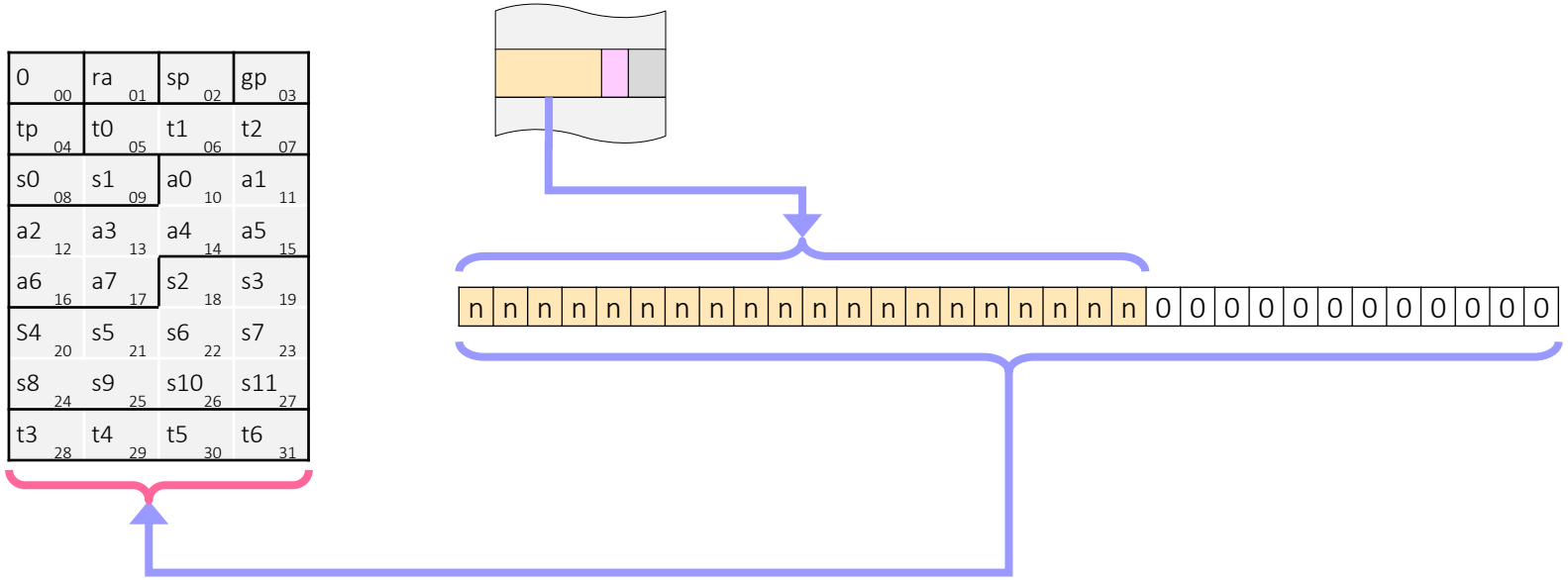
Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Formato U
lui rd, num	$rd_{31-12} \leftarrow num$ $rd_{11-0} \leftarrow 0$	n n n n n n n n n n n n n n n n n n n n d d d d d 0 1 1 0 1 1 1

num rd 0x37

load upper immediate

Carga un dato
en la parte alta de un registro



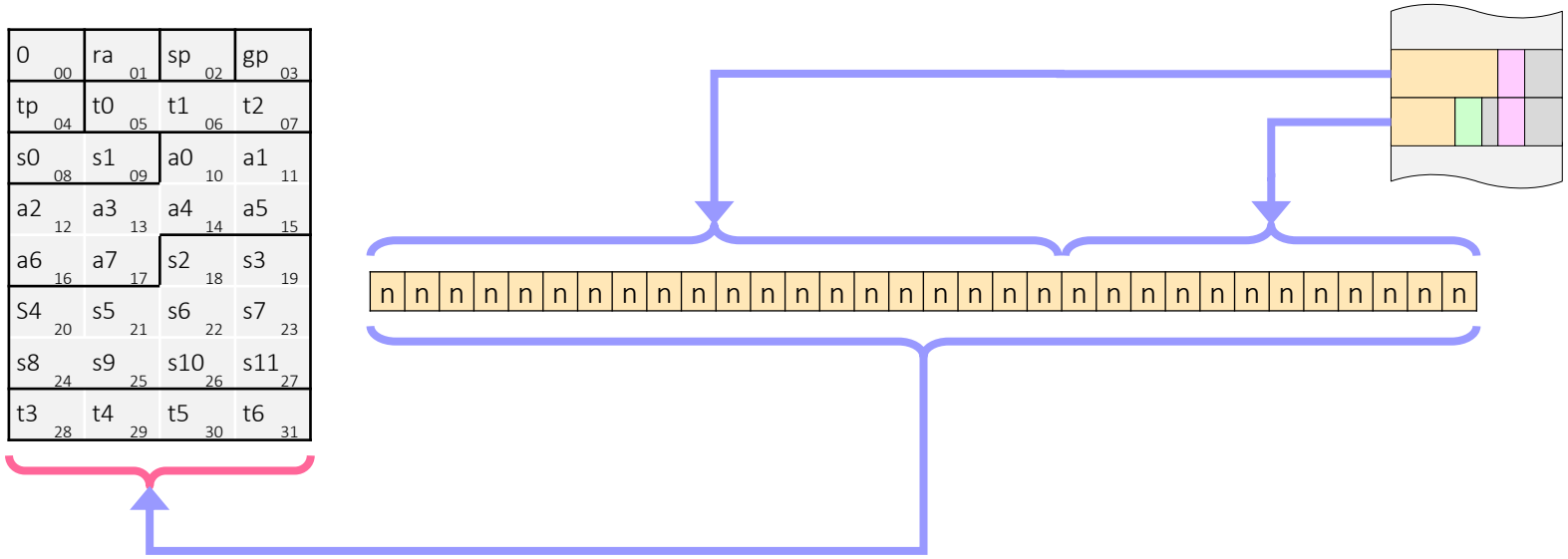
Instrucciones RISC-V

Transferencia de datos

Nemónico	Operación	Pseudo-instrucción
li rd, num	rd ← num_{12}	addi rd, zero, num_{11-0}
	rd ← num_{32}	lui rd, $num_{31:12}$ addi rd, rd, $num_{11:0}$

load immediate

Carga en un registro un valor de 32 bits

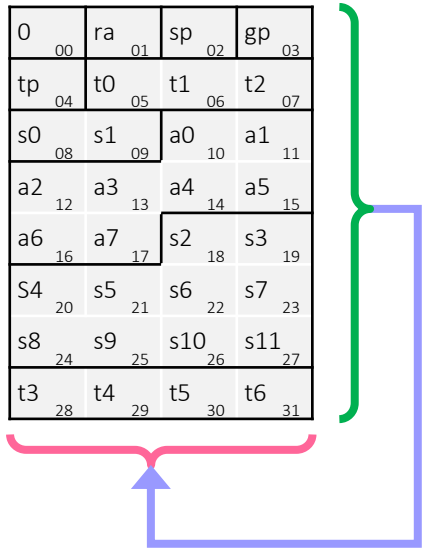


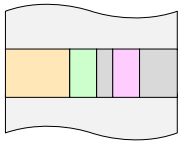
Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Pseudo-instrucción
mv rd, rs1	rd ← rs1	addi rd, rs1, 0

move

Copia el valor de un registro
a otro registro





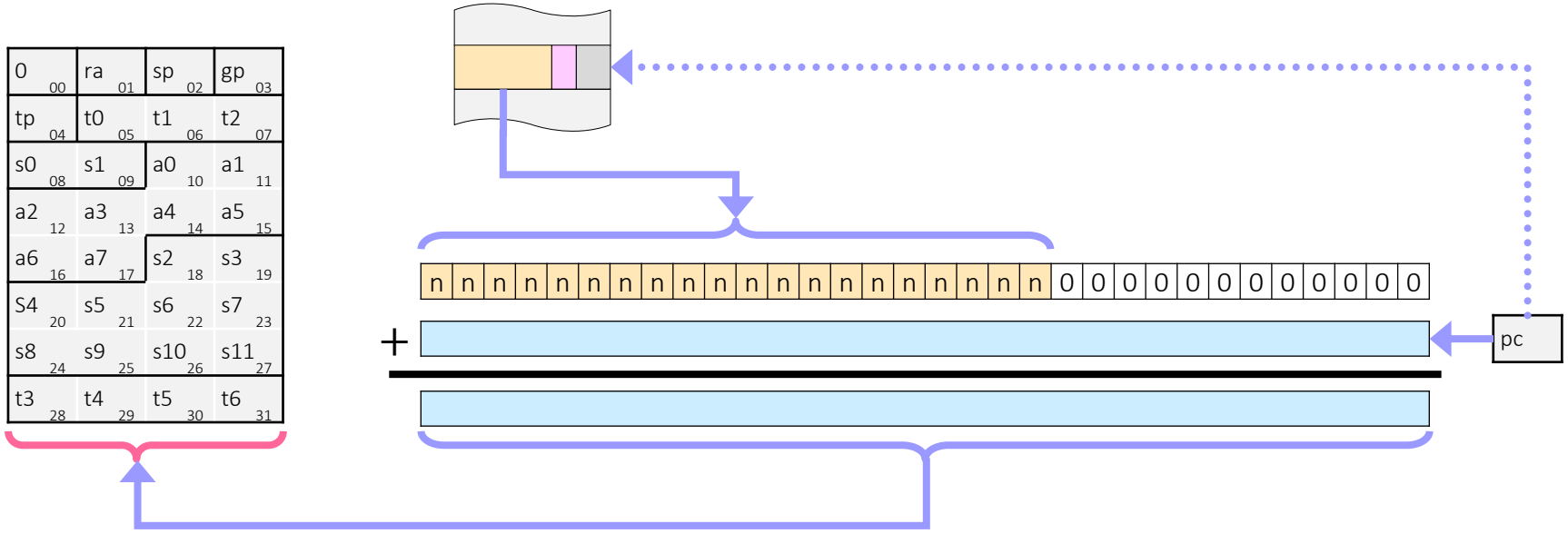
Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Formato U
auipc rd, num	rd ← PC + num × 2^{12}	n n n n n n n n n n n n n n n n n n n d d d d d 0 0 1 0 1 1 1

num rd 0x17

add upper immediate to PC

Carga un registro
con un valor relativizado al PC

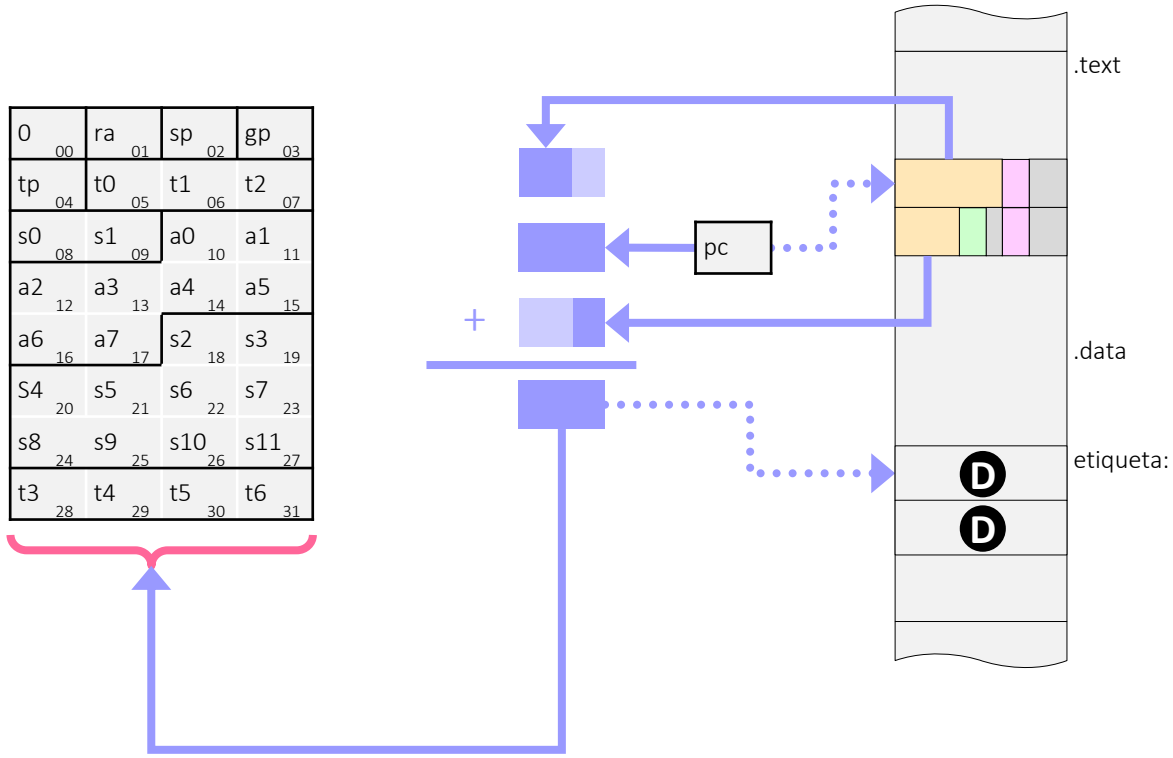


Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Pseudo-instrucción
la rd, etiqueta	rd ← $etiqueta_{32}$	auipc rd, $etiqueta_{31:12}$ addi rd, rd, $etiqueta_{31:0}$

load address

Carga en un registro
la dirección indicada por la etiqueta
calculada de modo que permite reubicar el
bloque (código junto con datos)

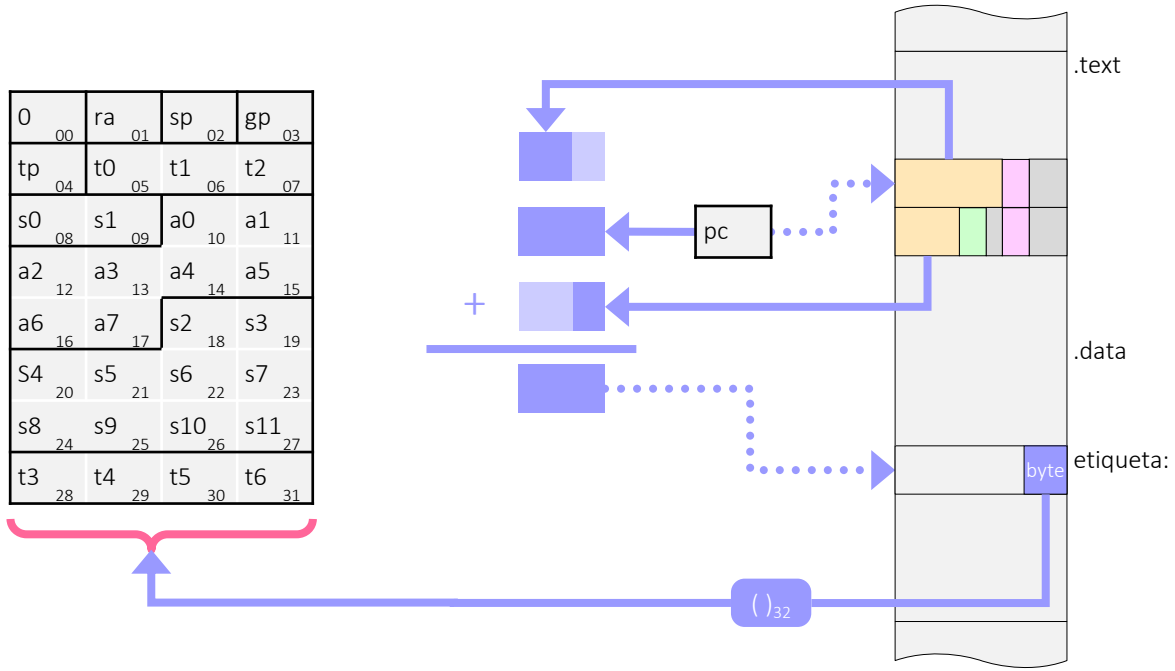


Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Pseudo-instrucción
lb rd, etiqueta	rd ← ([etiqueta$_{32}$]$_8$)$_{32}$	auipc rd, etiqueta$_{31:12}$ lb rd, etiqueta$_{11:0}$(rd)

load byte

Carga en un registro
un octeto de la dirección de memoria
indicada por la etiqueta,
al que se aplica extensión de signo

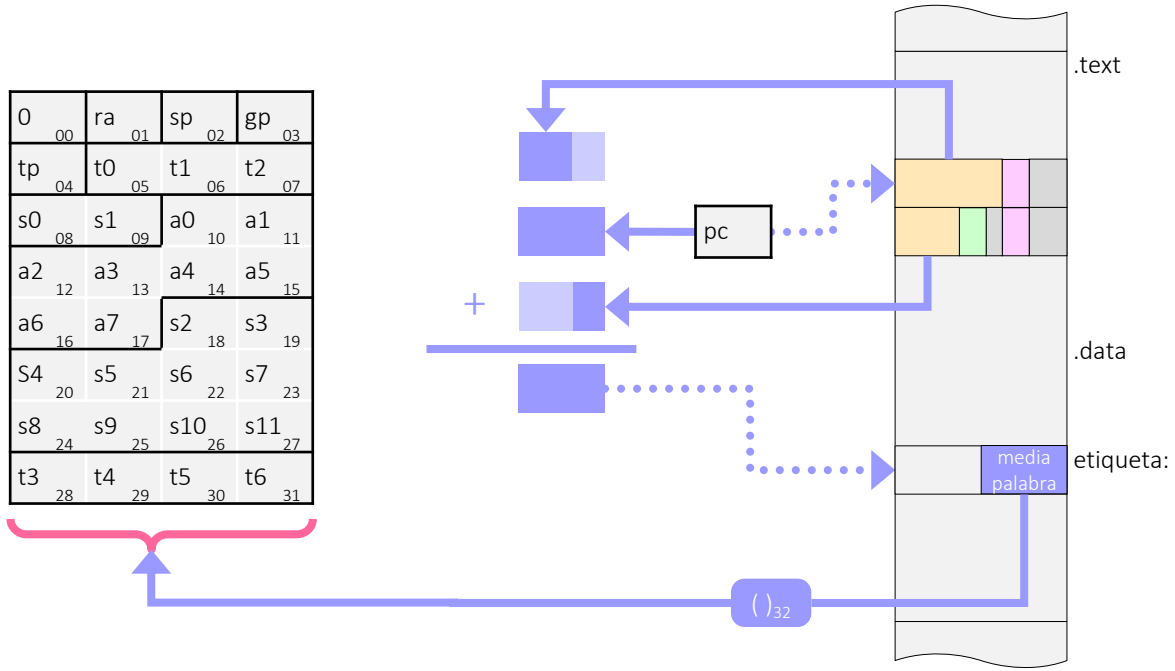


Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Pseudo-instrucción
lh rd, etiqueta	rd ← ([etiqueta$_{32}$]$_{16}$)$_{32}$	auipc rd, etiqueta$_{31:12}$ lh rd, etiqueta$_{11:0}$(rd)

load half

Carga en un registro
media palabra de la dirección de memoria
indicada por la etiqueta,
a la que se aplica extensión de signo

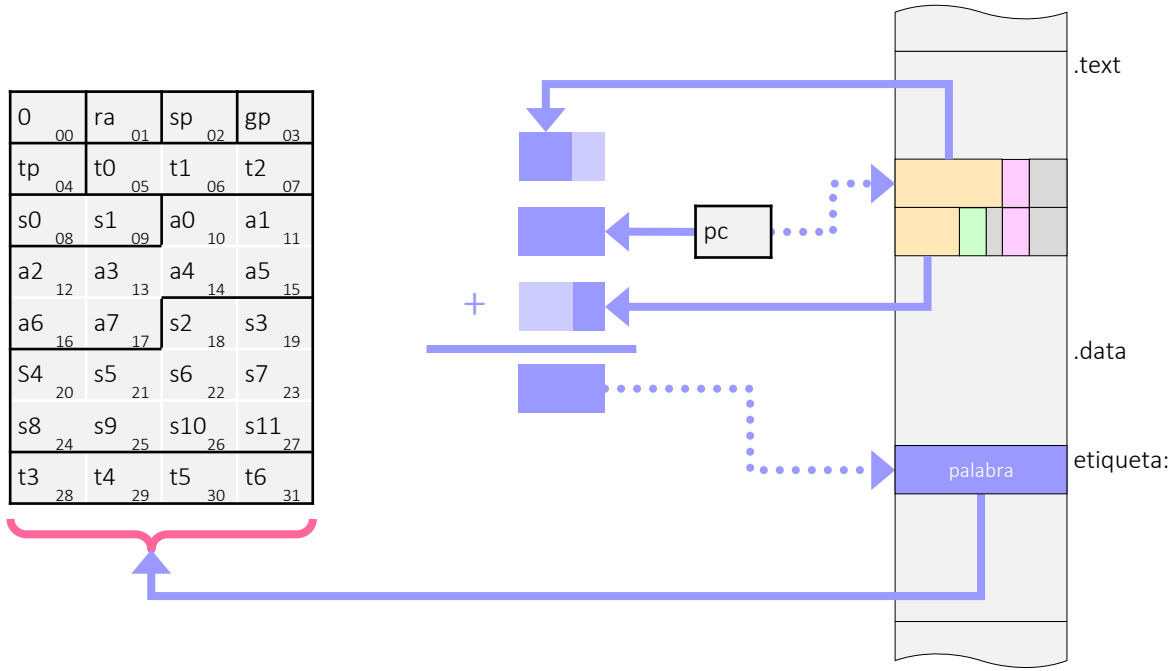


Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Pseudo-instrucción
lw rd, etiqueta	rd ← [etiqueta$_{32}$]	auipc rd, etiqueta$_{31:12}$ lw rd, etiqueta$_{11:0}$(rd)

load word

Carga en un registro
la palabra de la dirección de memoria
indicada por la etiqueta

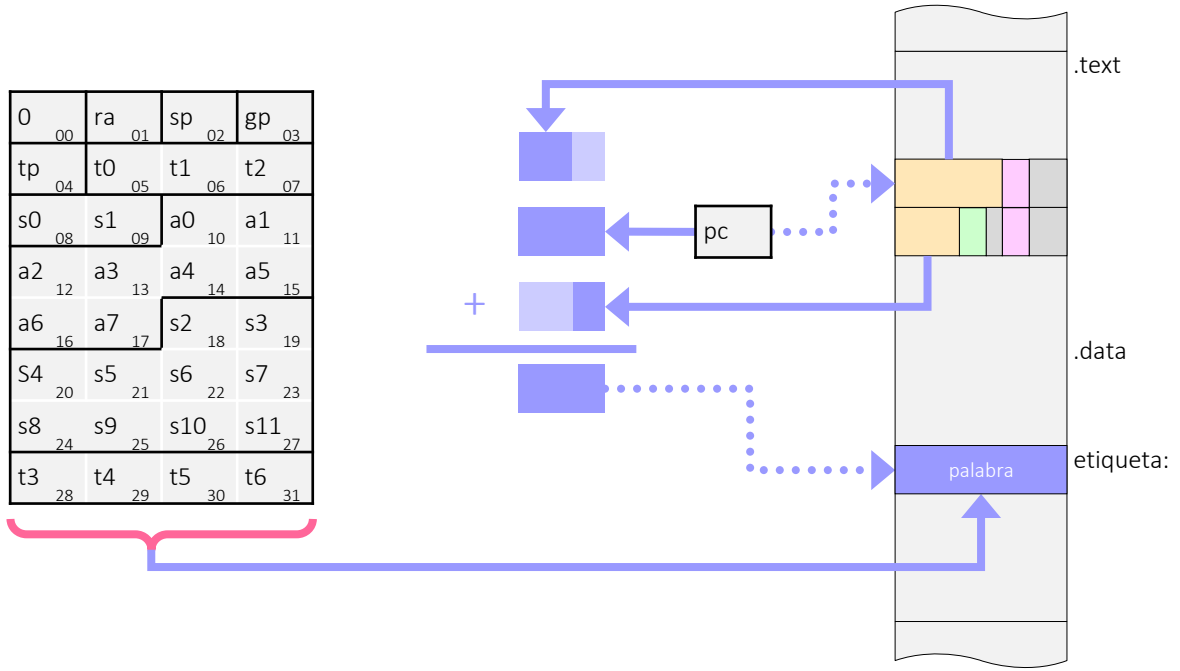


Instrucciones RISC-V
Transferencia de datos

Nemónico	Operación	Pseudo-instrucción
sw rs2, etiqueta, rs1	[etiqueta$_{32}$] ← rs2	auipc rs1, etiqueta$_{31:12}$ sw rd, etiqueta$_{11:0}$(rs1)

store word

Almacena un registro en la palabra de memoria indicada por la etiqueta



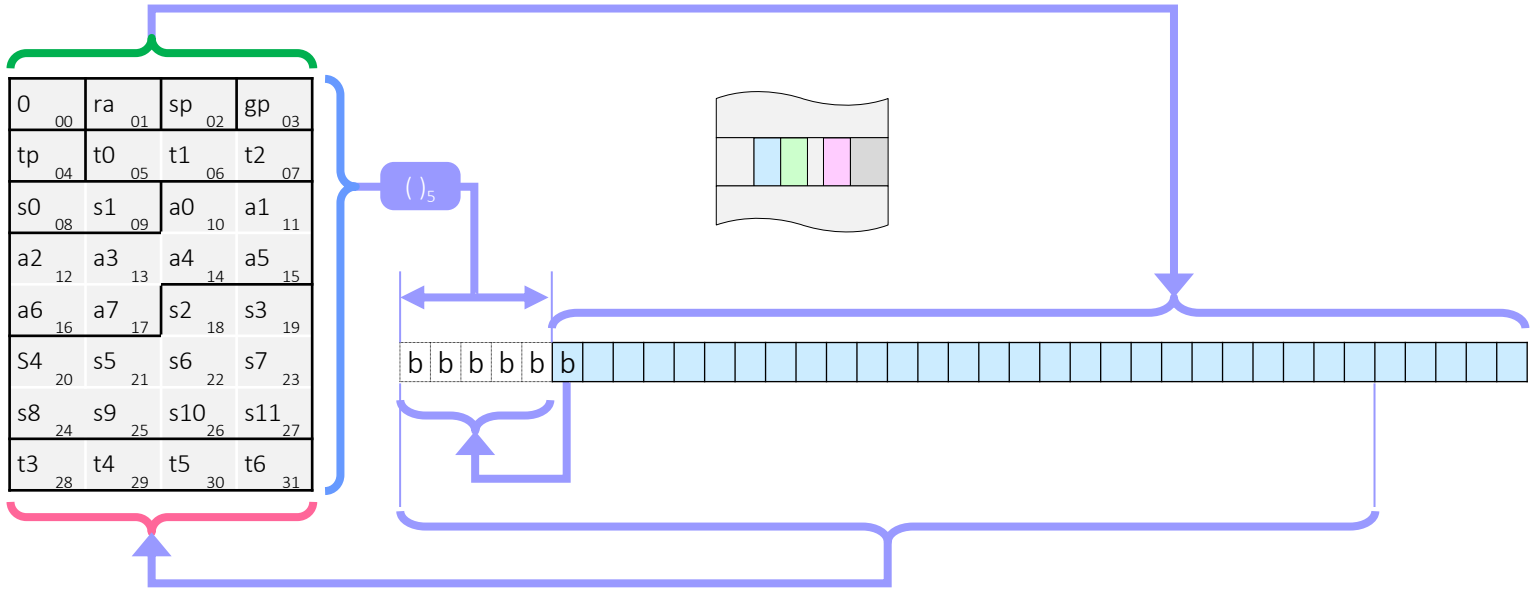
Instrucciones RISC-V

Operaciones de desplazamiento

Nemónico	Operación	Formato R
sra rd, rs1, rs2	rd ← rs1 / 2^{rs2} (en comp. a 2)	0 1 0 0 0 0 0 s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 1 0 1 d d d d d 0 1 1 0 0 1 1
		0x20 rs2 rs1 0x5 rd 0x33

shift right arithmetic

Carga en un registro el contenido de otro, previamente desplazado a la derecha tantos bits como indica un tercer registro. Los bits entrantes mantienen el signo.



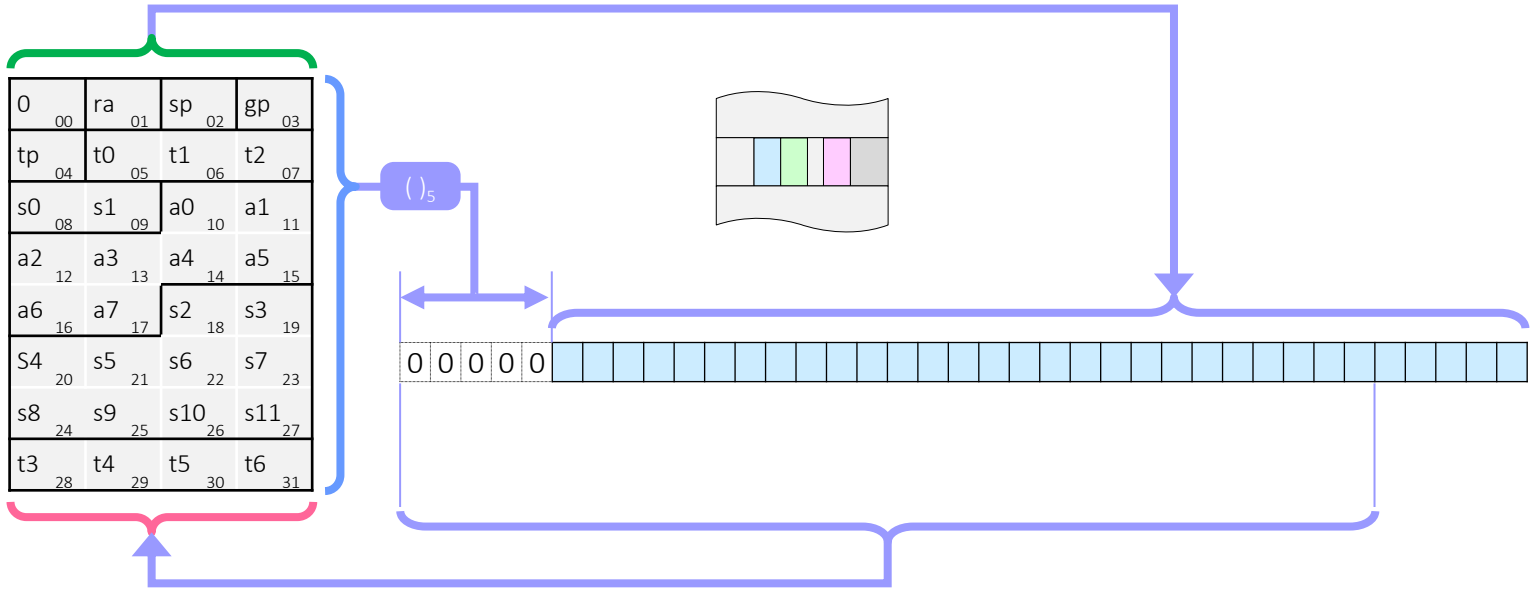
Instrucciones RISC-V

Operaciones de desplazamiento

Nemónico	Operación	Formato R
srl rd, rs1, rs2	rd ← rs1 / 2^{rs2} (en binario)	0 0 0 0 0 0 0 s2 s2 s2 s2 s2 s1 s1 s1 s1 s1 1 0 1 d d d d d 0 1 1 0 0 1 1
		0x0 rs2 rs1 0x5 rd 0x33

shift right logical

Carga en un registro el contenido de otro, previamente desplazado a la derecha tantos bits como indica un tercer registro. Los bits entrantes son ceros.

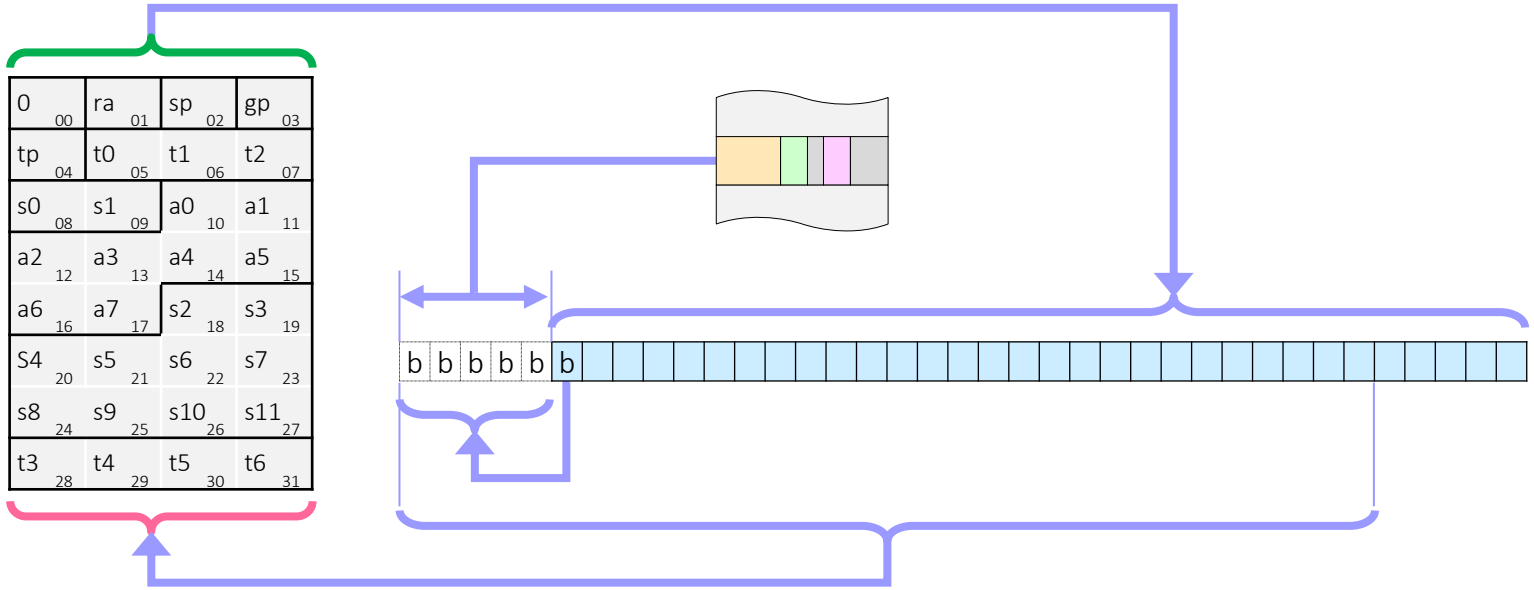


Instrucciones RISC-V
Operaciones de desplazamiento

Nemónico	Operación	Formato I
srai rd, rs1, num	rd ← rt / 2^{num} (en comp. a 2)	0 1 0 0 0 0 0 n n n n n s1 s1 s1 s1 s1 1 0 1 d d d d d 0 0 1 0 0 1 1
		num rs1 0x5 rd 0x13

shift right arithmetic immediate

Carga en un registro el contenido de otro, previamente desplazado a la derecha tantos bits como indica un argumento. Los bits entrantes mantienen el signo.



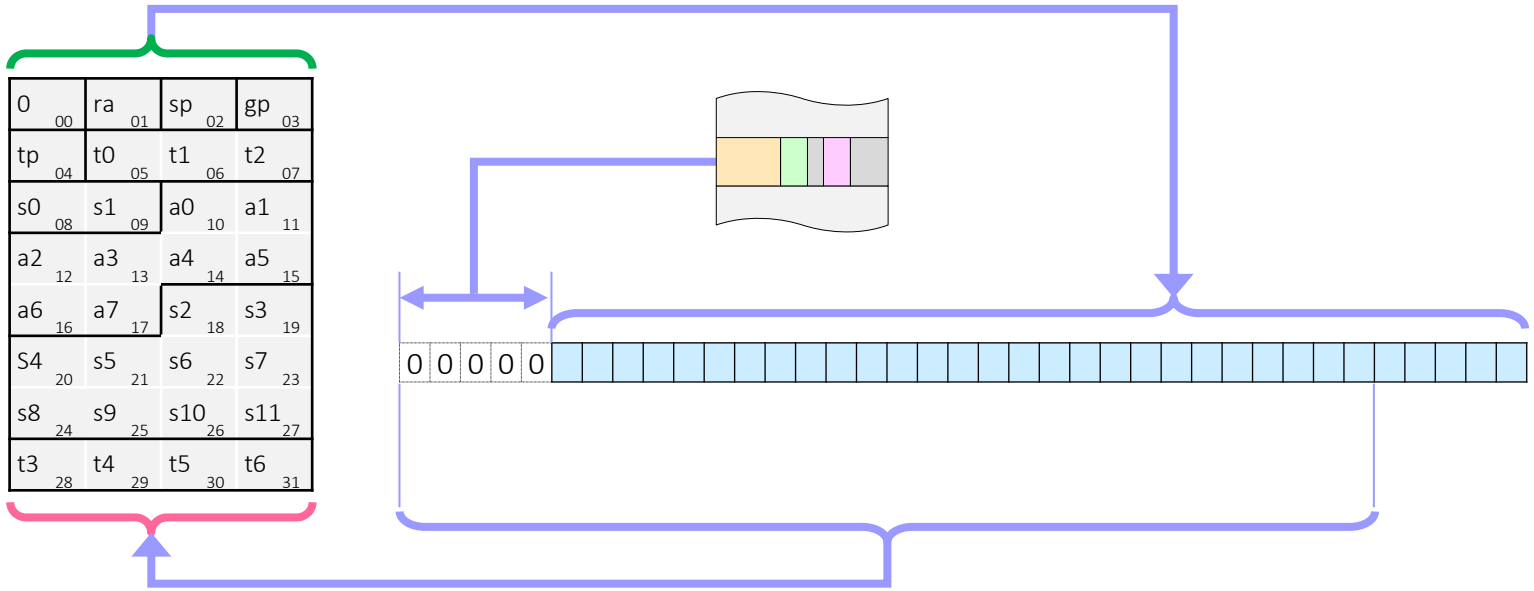
Instrucciones RISC-V

Operaciones de desplazamiento

Nemónico	Operación	Formato I
srli rd, rs1, num	rd ← rt / 2^{num} (en binario)	0 0 0 0 0 0 0 n n n n n s1 s1 s1 s1 s1 1 0 1 d d d d d 0 0 1 0 0 1 1
		num rs1 0x5 rd 0x13

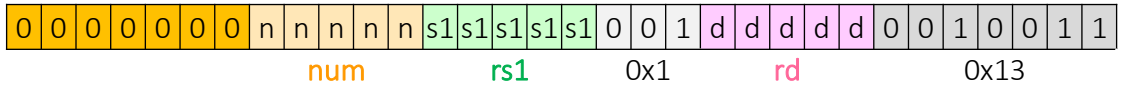
shift right logical immediate

Carga en un registro el contenido de otro, previamente desplazado a la derecha tantos bits como indica un argumento. Los bits entrantes son ceros.



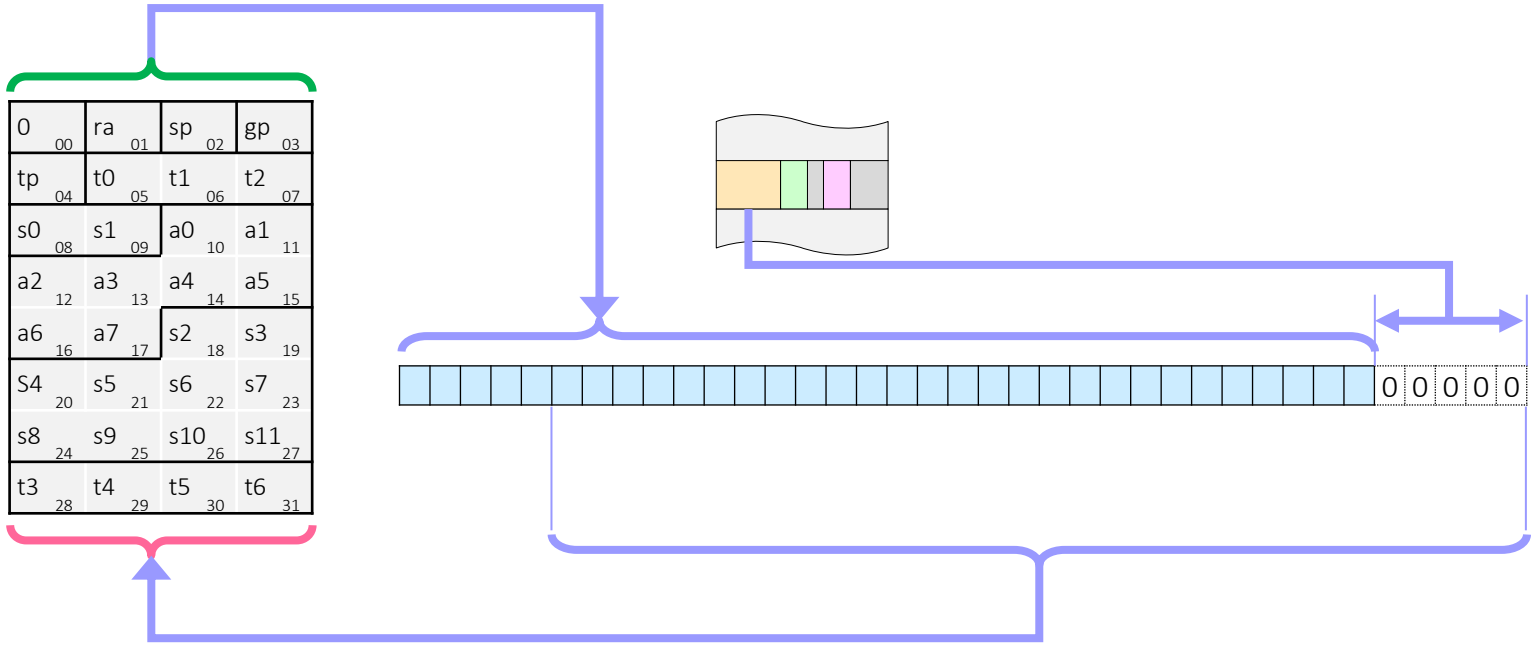
Instrucciones RISC-V
Operaciones de desplazamiento

Nemónico	Operación	Formato I
slli rd, rs1, num	rd ← rt × 2^{num}	



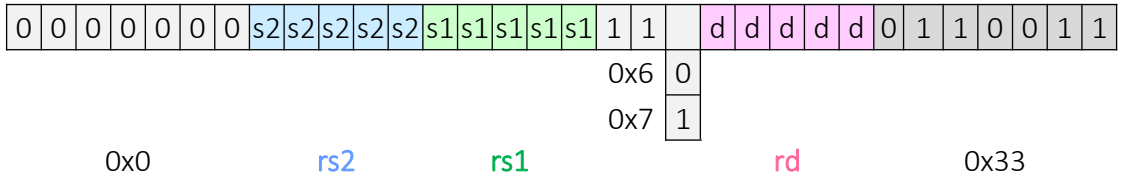
shift left logical immediate

Carga en un registro el contenido de otro, previamente desplazado a la izquierda tantos bits como indica un argumento. Los bits entrantes son ceros.



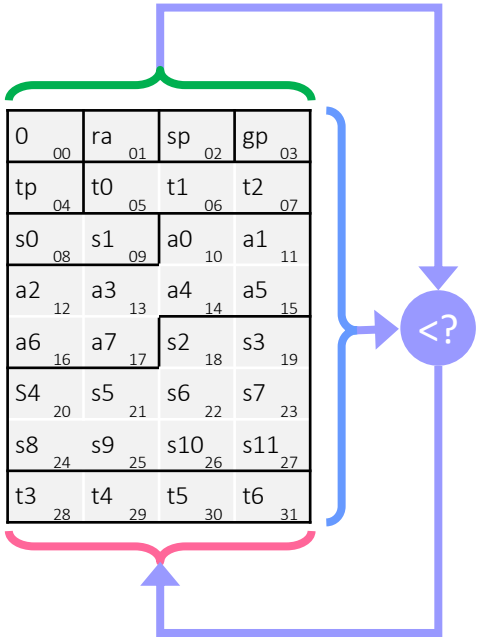
Instrucciones RISC-V
Operaciones de comparación

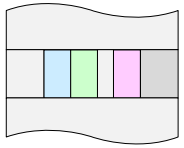
Nemónico	Operación	Formato R
slt rd, rs1, rs2	rd ← 1 si rs1 < rs2 ; 0 si no en complemento a dos	
sltu rd, rs1, rs2	en binario natural	



set if less than

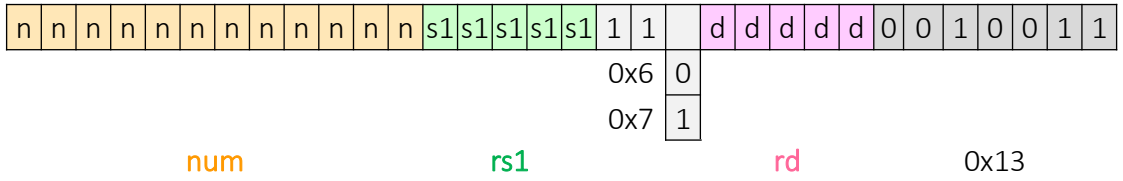
Verifica si el valor de un registro es menor que el valor de otro





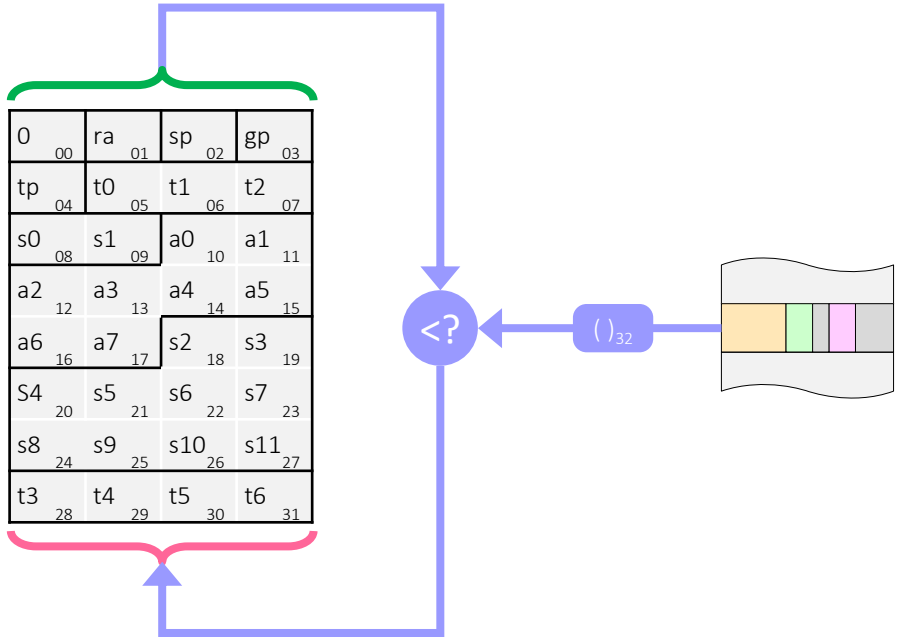
Instrucciones RISC-V
Operaciones de comparación

Nemónico	Operación	Formato I
	rd ← 1 si rs < num ; 0 si no	
slti rd, rs1, num	en complemento a dos	
sltiu rd, rs1, num	en binario natural	



set if less than immediate

Verifica si el valor de un registro es menor que una cantidad dada

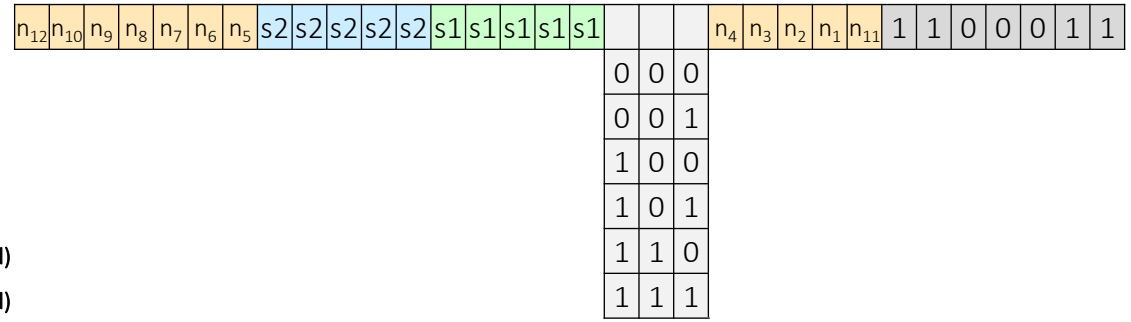


Instrucciones RISC-V

Saltos condicionales

Nemónico		Operación
		pc ← pc + offset
beq	rs1, rs2, num	bifurca si rs1 = rs2
bne	rs1, rs2, num	bifurca si rs1 ≠ rs2
blt	rs1, rs2, num	bifurca si rs1 < rs2
bge	rs1, rs2, num	bifurca si rs1 ≥ rs2
bltu	rs1, rs2, num	bifurca si rs1 < rs2 (unsigned)
bgeu	rs1, rs2, num	bifurca si rs1 ≥ rs2 (unsigned)

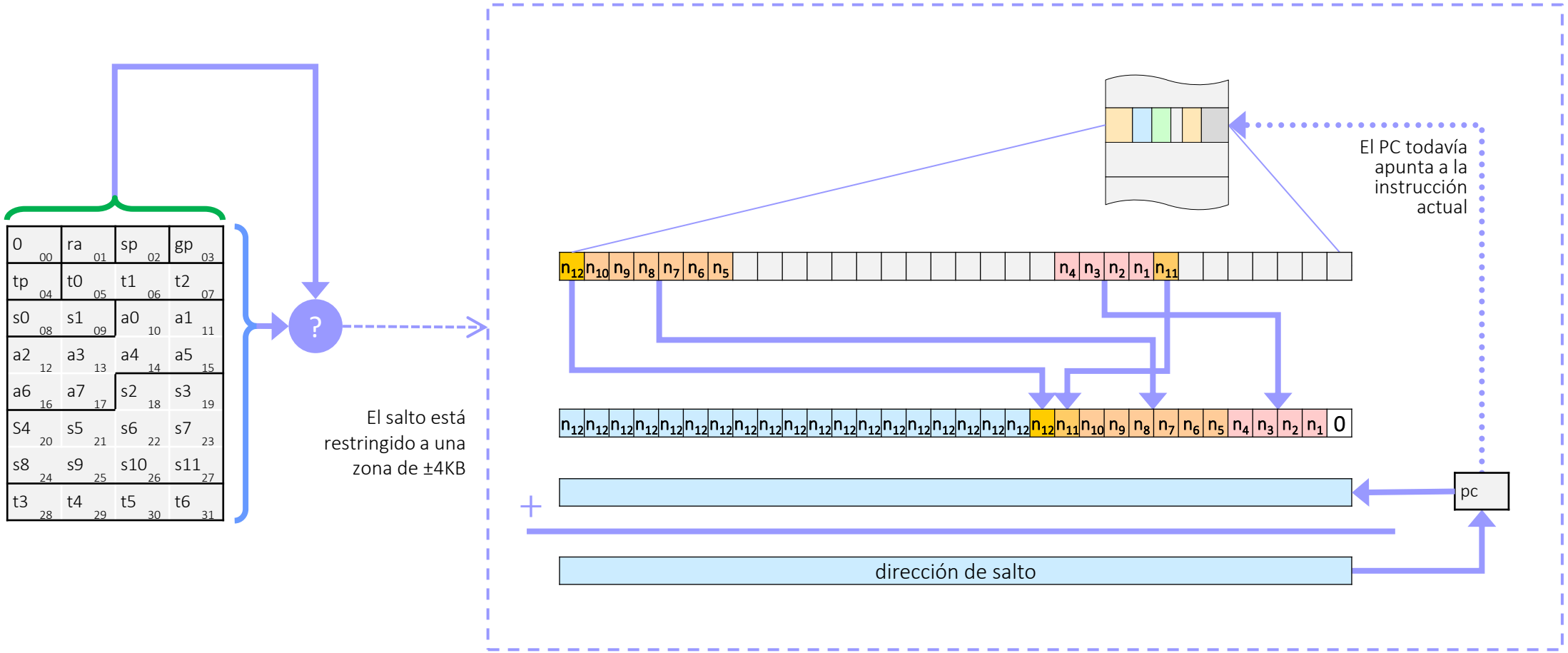
Formato B



Nemónico		Operación	Pseudo-instrucción		
beqz	rs1, num	bifurca si rs1 = 0	beq	rs1, zero, num
bnez	rs1, num	bifurca si rs1 ≠ 0	bne	rs1, zero, num
bltz	rs1, num	bifurca si rs1 < 0	blt	rs1, zero, num
bgtz	rs1, num	bifurca si rs1 > 0	blt	zero, rs1, num
blez	rs1, num	bifurca si rs1 ≤ 0	bge	rs1, zero, num
bgez	rs1, num	bifurca si rs1 ≥ 0	bge	zero, rs1, num
ble	rs1, rs2, num	bifurca si rs1 ≤ rs2	bge	rs2, rs1, num
bgt	rs1, rs2, num	bifurca si rs1 > rs2	blt	rs1, rs2, num
bleu	rs1, rs2, num	bifurca si rs1 ≤ rs2 (unsigned)	bgeu	rs2, rs1, num
bgtu	rs1, rs2, num	bifurca si rs1 > rs2 (unsigned)	bltu	rs1, rs2, num

Instrucciones RISC-V

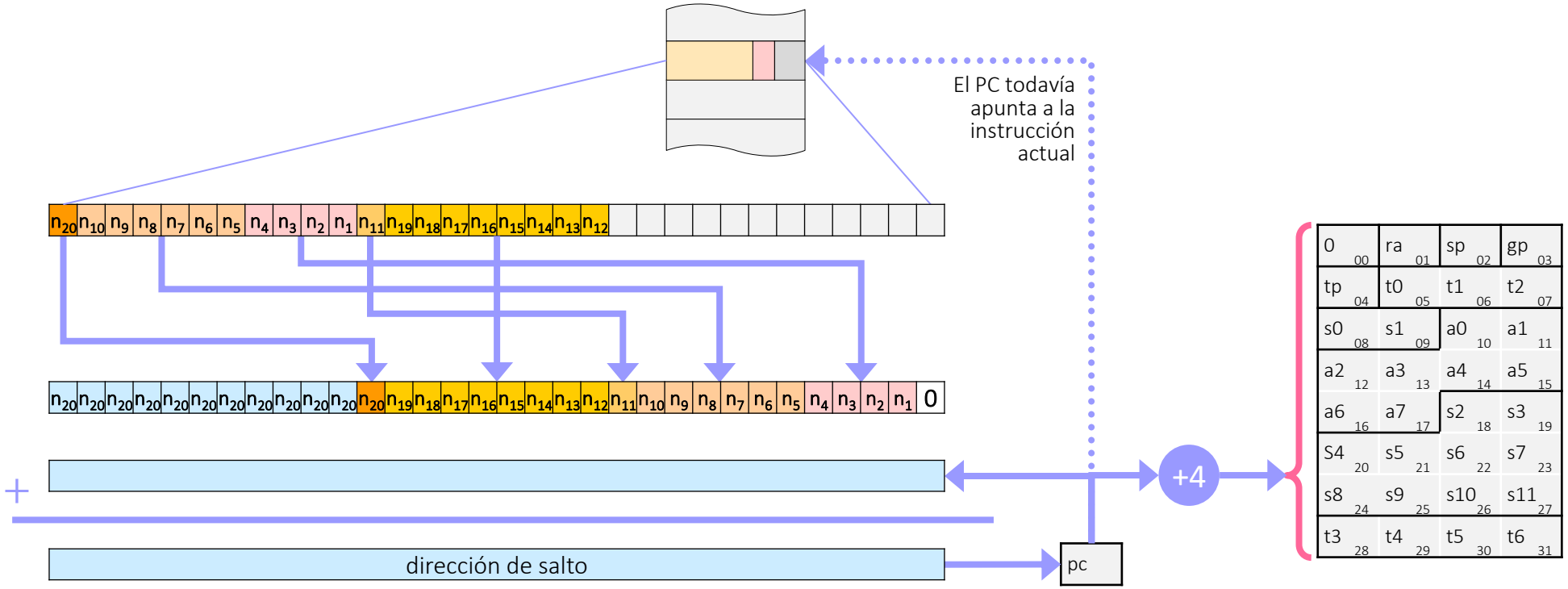
Saltos condicionales



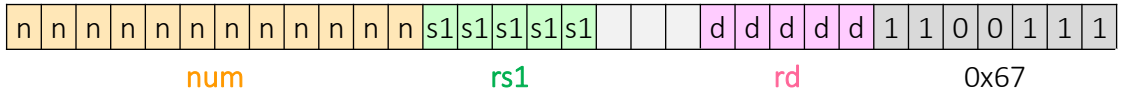
Instrucciones RISC-V

Saltos incondicionales

Nemónico	Operación	Formato J
jal rd, num	rd ← pc + 4 pc ← pc + offset(num)$_{32}$	num · rd · 0x67



Instrucciones RISC-V
Saltos incondicionales

Nemónico		Operación	Formato I
jalr	rd, rs1, num	rd ← pc + 4 pc ← rs1 + $(num)_{32}$	

Nemónico		Operación	Pseudo-instrucción	
j	etiqueta	salto	jal	zero, etiqueta
jr	rs1	salto mediante registro	jalr	zero, rs1, 0
jal	etiqueta		jal	ra, etiqueta
call	etiqueta	llamada cercana	jal	ra, etiqueta
call	etiqueta	llamada lejana	auipc	ra, $etiqueta_{31:12}$
			jalr	ra, ra, $etiqueta_{11:0}$
ret		retorno cercano	jalr	zero, ra, 0

Instrucciones RISC-V

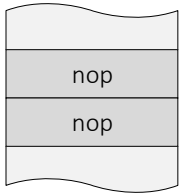
Instrucciones misceláneas

Nemónico	Operación	Pseudo-instrucción
nop	no operar	addi zero, zero, 0

no operation

Instrucción que ocupa lugar y se ejecuta, pero no hace nada relevante.

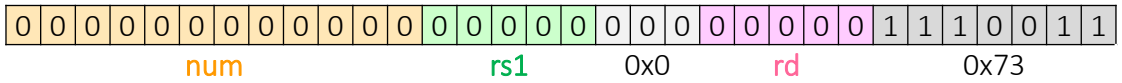
Sirve para generar huecos en el código de un programa, que podrían completarse posteriormente.



Instrucciones RISC-V
Instrucciones misceláneas

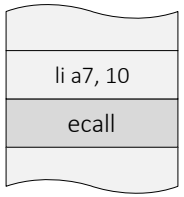
Nemónico	Operación
ecall	llamada a servicio del SO

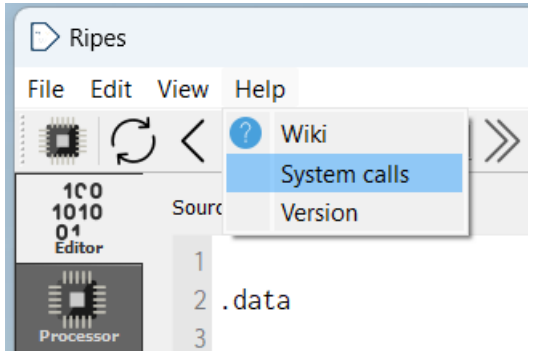
Formato I



environment call

Realiza una solicitud
(indicada en el registro a7)
al entorno de soporte de ejecución





Servicio	a7	Descripción	Argumentos	Resultado
PrintInt	1	Prints an integer	a0 = integer to print	N/A
PrintFloat	2	Prints a floating point number	fa0 = float to print	N/A
PrintString	4	Prints a null-terminated string to the console	a0 = the address of the string	N/A
Exit	10	Exits the program with code 0	N/A	N/A
PrintChar	11	Prints an ascii character	a0 = character to print (only lowest byte is considered)	N/A
ReadChar	12	Reads a character from input console	N/A	a0 = the character
GetCWD	17	Writes the path of the current working directory into a buffer	a0 = the buffer to write into a1 = the length of the buffer	a0 = -1 if the path is longer than the buffer
Time_ms	30	Get the current time (milliseconds since 1 January 1970)	N/A	a0 = low order 32 bits a1=high order 32 bits
Cycles	31	Get number of cycles elapsed since program start	N/A	a0=low 32 bits of cycles elapsed a1= high 32 bits of cycles elapsed
PrintIntHex	34	Prints an integer (in hexdecimal format left-padded with zeroes)	a0 = integer to print	N/A
PrintIntBinary	35	Prints an integer (in binary format left-padded with zeroes)	a0 = integer to print	N/A
PrintIntUnsigned	36	Prints an integer (unsigned)	a0 = integer to print	N/A
Close	57	Close a file	a0 = the file descriptor to close	N/A
LSeek	62	Seek to a position in a file	a0 = the file descriptor a1 = the offset for the base a2 is the begining of the file (0), the current position (1), or the end of the file (2)}	a0 = the selected position from the beginning of the file or -1 is an error occurred
Read	63	Read from a file descriptor into a buffer	a0 = the file descriptor a1 = address of the buffer a2 = maximum length to read	a0 = the length read or -1 if error
Write	64	Write to a file descriptor from a buffer	a0 = the file descriptor a1 = the buffer address a2 = the length to write	a0 = the number of characters written
FStat	80	Is used to determine information about a file based on its file descriptor	a0=the file descriptor a1=pointer to a struct stat	a0=returns -1 if an error occurred
Exit2	93	Exits the program with a code	a0 = the number to exit with	N/A
brk	214	Change the locations of the program break, witch the end of the process's data segment	a0=sets the end of the data segment to the specified address	a0=On success, brk() returns zero. On error, -1 is returned, and errno is set to ENOMEM
Open	1024	Opens a file from a path	a0 = Null terminated string for the path a1 = flags	a0 = the file descriptor or -1 if an error occurred