



Día, Fecha:	12 de Abril de 2024
Hora de inicio:	16:30 a 18:10

0781 ORGANIZACION DE LENGUAJES Y COMPILADORES 2

Daniel Enrique Santos Godoy



PRIMER SEMESTRE - 2023

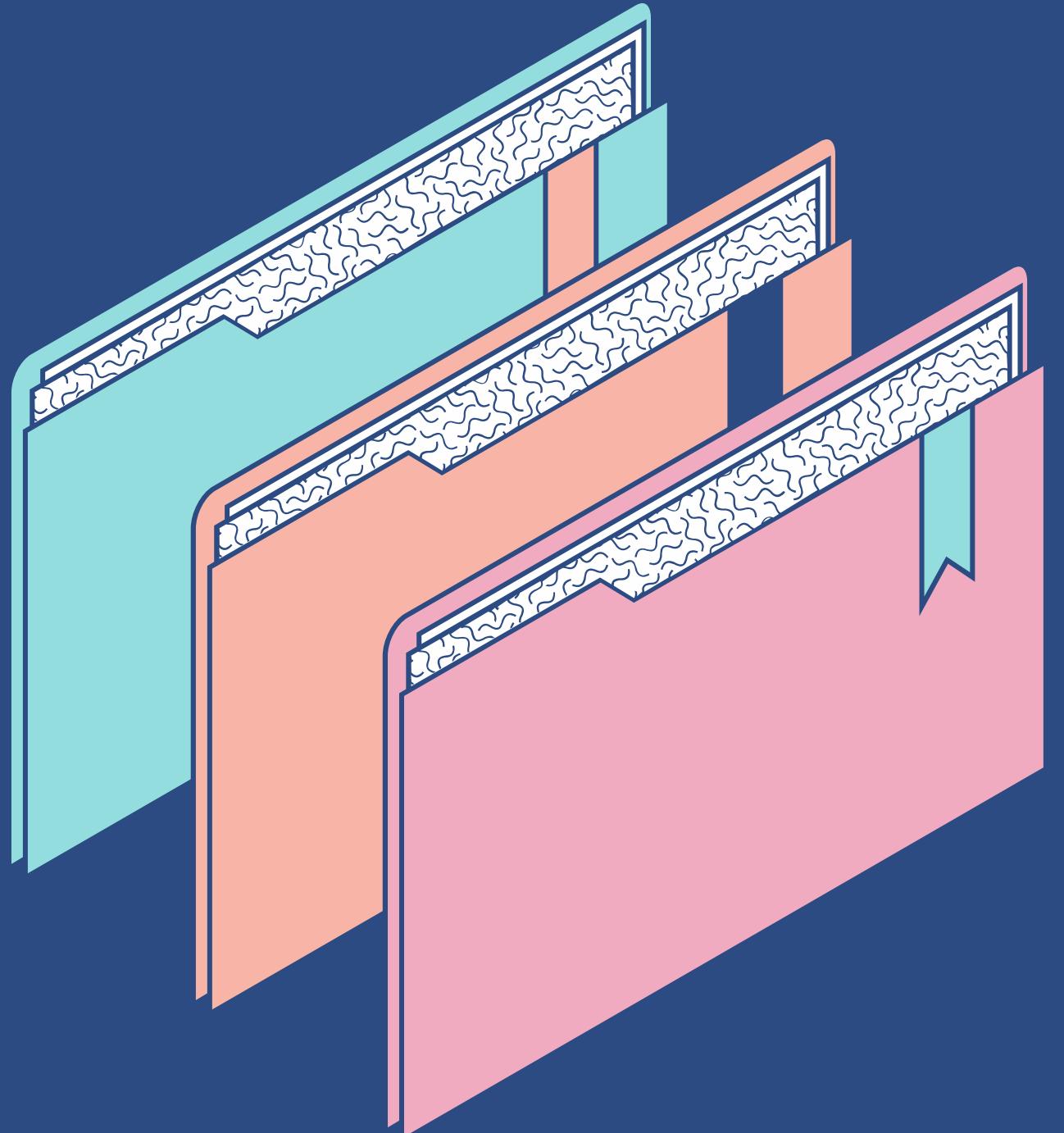
Organización de Lenguajes y Compiladores 2

Clase 8 - RISC-V

Agenda

TEMAS CLAVE QUE SE
DEBATIRÁN EN ESTA
PRESENTACIÓN

- ¿Dudas del proyecto?
- Estructuras de Control
- Arrays
- Ejemplos Prácticos
- ¿Dudas?





Dudas del Proyecto



Estructuras de Control

Las estructuras de control en lenguaje ensamblador RISC-V, como en cualquier otro lenguaje de programación, permiten dirigir el flujo de la ejecución del programa. En RISC-V, estas estructuras de control no existen como tales instrucciones específicas, pero se implementan usando una combinación de instrucciones de salto y comparación.

Funciones de Control

- BEQ (Branch if Equal): Salta si los dos registros son iguales.
- BNE (Branch if Not Equal): Salta si los dos registros no son iguales.
- BLT (Branch if Less Than): Salta si el primer registro es menor que el segundo (considerando signo).
- BGE (Branch if Greater or Equal): Salta si el primer registro es mayor o igual que el segundo (considerando signo).
- BLTU (Branch if Less Than Unsigned): Similar a BLT, pero para números sin signo.
- BGEU (Branch if Greater or Equal Unsigned): Similar a BGE, pero para números sin signo.

BEQ (Branch if Equal)

Sintaxis: beq rs1, rs2, label

Descripción: Esta instrucción compara dos registros, rs1 y rs2. Si los valores en estos registros son iguales, el flujo de ejecución salta a la dirección marcada por label.

Uso: Se utiliza para implementar estructuras de control que dependen de la igualdad, como un bucle o una condición if

```
(ML, false);  
    PHP_VERSION, ">")) {  
5.2", PHP_VERSION, ">");  
reater is required!!!");  
d("pcre")) {  
requires the pcre extension to php in c  
oot . /includes/autoload.php';  
ion . /config.php';  
oot . /config.php';  
_CONFIG_FILE') || !defined('PSI_DEBUG');  
template('/templates/html/error_config.  
atch();
```

BNE(Branch if Not Equal)

Sintaxis: bne rs1, rs2, label

Descripción: Compara dos registros, rs1 y rs2. Si los valores son diferentes, entonces el flujo de ejecución se desvía al label especificado.

Uso: Útil para ciclos de repetición donde se espera un cambio de valor, o para verificar si dos valores no son iguales antes de proceder con cierta parte del código.



BLT (Branch if Less Than)

Sintaxis: blt rs1, rs2, label

Descripción: Compara dos registros interpretados como enteros con signo. Si el valor en rs1 es menor que el valor en rs2, entonces salta al label.

Uso: Puede ser usada en bucles o condiciones donde se requiere comparar magnitudes, como en la ordenación de datos o en decisiones basadas en valores numéricos.

(ML, false);
5.2", PHP_VERSION, ">")) {
 greater is required!!!";
}
d("pcre"));
requires the pcre extension to php in c
oot ./includes/autoload.inc.php';
ion ./config.php';
oot ./config.php';
defined('PSI_DEBUG')
_CONFIG_FILE') || !defined('PSI_DEBUG')
template('/templates/html/error_config.
atch();

BGE (Branch if Greater or Equal)

Sintaxis: bge rs1, rs2, label

Descripción: Compara dos registros como enteros con signo. Si el valor en rs1 es mayor o igual que el valor en rs2, se realiza un salto al label.

Uso: Ideal para verificaciones de límites superiores en condiciones y bucles, donde necesitas asegurar que un valor sigue siendo superior o igual a otro.

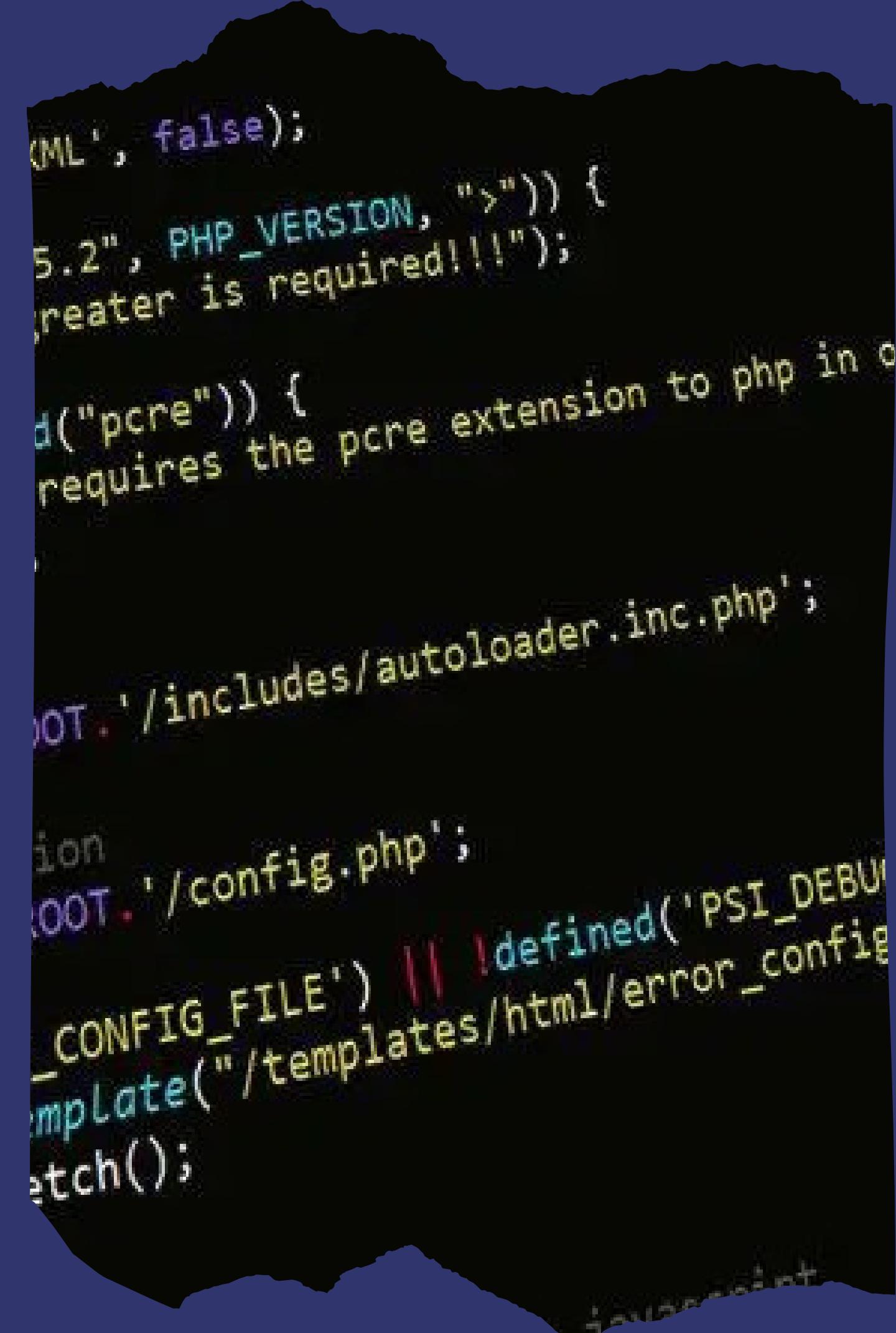


BLTU (Branch if Less Than Unsigned)

Sintaxis: bltu rs1, rs2, label

Descripción: Similar a BLT, pero compara los valores de rs1 y rs2 como números sin signo. Esto es útil cuando trabajas con números donde no es relevante o posible un valor negativo (como índices de arreglo).

Uso: Usado en situaciones donde los datos son inherentemente no negativos, como direcciones de memoria, tamaños de objetos, o cuando manipulas datos que deben interpretarse como bits crudos en lugar de como números.



BGEU (Branch if Greater or Equal Unsigned)

Sintaxis: bgeu rs1, rs2, label

Descripción: Funciona como BGE, pero compara rs1 y rs2 como enteros sin signo.

Uso: Aplicable en contextos donde se comparan valores que no pueden ser negativos, asegurando que se trate de comparaciones de magnitudes en un contexto sin signo.

```
(ML, false);
5.2", PHP_VERSION, ">")) {
    greater is required!!!";
}
d("pcre")) {
    requires the pcre extension to php in order to use it.
}
OOT."/includes/autoload.inc.php';
ion
OOT."/config.php';
defined('PSI_DEBUG') || !defined('PSI_CONFIG_FILE')
    _template('/templates/html/error_config');
    catch();
```



Arrays

Declaracion

En RISCV podemos declarar un Array dentro de la sección .data del programa.

```
.data  
array: .word 10, 20, 30, 40, 50 # Un array de 5 elementos
```



Arrays

Acceso

Para acceder a los elementos del array, se debe calcular las direcciones de cada elemento utilizando la dirección base del array y el desplazamiento correspondiente al índice del elemento. En RISC-V, cada word tiene 4 bytes, por lo que si se quiere acceder al elemento i -ésimo del array, el desplazamiento será $4*i$.

```
la x10, array    # Cargar la dirección base del array en x10  
lw x5, 8(x10)   # Cargar el tercer elemento en x5 (2*4 = 8 bytes de desplazamiento)
```



Arrays

Modificar un Elemento

Para modificar un elemento, se debe utilizar una instrucción de almacenamiento con la dirección calculada de manera similar:

```
li x6, 100      # Cargar el valor 100 en el registro x6  
sw x6, 8(x10)  # Almacenar el valor de x6 en el tercer elemento del array
```



Arrays

Iterar Sobre un Array

Para iterar sobre un array se debe utilizar un bucle, con las instrucciones de control que se vieron anteriormente. (Se vera en el ejemplo)

Saltos

JAL (Jump and Link)

Sintaxis: jal rd, offset

Descripción: Esta instrucción realiza un salto incondicional a la dirección especificada por el offset desde la posición actual del contador de programa (PC). Además, guarda la dirección de retorno (la dirección de la siguiente instrucción ejecutable después de jal) en el registro rd. Es típicamente usada para llamadas a funciones.

Saltos

JALR (Jump and Link Register)

Sintaxis: jalr rd, rs1, offset

Descripción: Realiza un salto incondicional a la dirección calculada como la suma de un offset y el contenido del registro rs1. Guarda la dirección de retorno en rd. Esta instrucción es útil para realizar retornos de funciones o saltos a direcciones almacenadas en registros.

Saltos

J (Jump)

Sintaxis: j label

Descripción: Es una pseudo-instrucción en RISC-V, la cual es en realidad una forma simplificada de jal donde no se guarda la dirección de retorno (típicamente, jal x0, label se puede escribir como j label). Realiza un salto incondicional a la etiqueta especificada.

Saltos

JR (Jump Register)

Sintaxis: jr rs1

Descripción: Otra pseudo-instrucción, esencialmente una forma simplificada de jalr sin guardar la dirección de retorno (equivalente a jalr x0, rs1, 0). Salta a la dirección contenida en el registro rs1.

Saltos

RET (Return)

Sintaxis: ret

Descripción: Esta es una pseudo-instrucción utilizada para retornar de una subrutina. Es equivalente a jalr x0, x1, 0, asumiendo que x1 (el registro de enlace) contiene la dirección de retorno.

A collage of various school-related illustrations on a dark blue background. It includes a top-down view of a notebook with a wavy pattern, a pencil case with three pencils, a stack of books, a laptop keyboard, a potted plant with long green and pink leaves, and a small white device with a screen and buttons.

PRIMER SEMESTRE - 2023

Gracias por su
atención..

