CS3402
Compiladores
Julio Eduardo Yarasca Moscol

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Compiladores: Proyecto Final

Instrucciones:

1. El objetivo del proyecto es desarrollar un compilador que genere código en lenguaje ensamblador x86-64 para un lenguaje de programación asignado.

2. Los grupos deben estar conformados por un máximo de 3 integrantes.

3. El entregable estará compuesto por:

   - Código fuente en C++, estructurado de la siguiente manera: `scanner`, `parser`, `visitors`, casos de prueba (`test cases`) y un archivo `Makefile`.
   - Presentación.
   - Reporte.

4. Requisitos del código:

   - El grupo que no respete la estructura exigida recibirá una calificación de cero.
   - Solo se otorgará puntaje si el compilador genera código ensamblador que incluya: declaración de variables, expresiones aritméticas, sentencias de control (condicionales y bucles) y funciones.

5. Cada grupo debe seleccionar dos extensiones de la siguiente lista e implementarlas completamente:

   - Float type
   - Unsigned int
   - Long int
   - Memoria dinámica
   - Strings
   - Arrays
   - Struct
   - Punteros

   Además, se deben implementar sus operaciones y métodos asociados.

6. Casos de prueba requeridos:

   - 3 test cases de declaración de variables
   - 3 test cases de expresiones
   - 6 test cases de sentencias de control selectivo
   - 3 test cases de funciones
   - 5 test cases para la primera extensión elegida
   - 5 test cases para la segunda extensión elegida

7. Se debe elaborar un `Makefile` (o un script en Python) que:
   - Compile el proyecto.
   - Ejecute automáticamente todos los test cases.
   - Genere archivos de salida en código ensamblador (`.s`).
8. El informe y la presentación deben enfocarse exclusivamente en las extensiones implementadas. Se deben incluir:
   - Clases elaboradas
   - Operaciones implementadas
   - Estructura de los visitors y diseño.
9. Puntaje adicional: Los grupos que implementen una interfaz gráfica que permita escribir código en el lenguaje definido, visualizar el código ensamblador generado y ejecutar el programa recibirán una bonificación.
10. La calificación total será sobre 20 puntos, según la siguiente rúbrica:

Implementación base	8 pts
Implementación de extensiones	6 pts
Reporte	3 pts
Presentación oral	3 pts
Interfaz gráfica (bonificación)	+2 pts

11. La fecha de entrega es el lunes 30 de junio hasta las 11:59 p.m.
12. Las presentaciones tendrán una duración de 30 minutos: 20 minutos de exposición y 10 minutos para preguntas.

A. Ejemplos sencillos

A.1. Lenguaje Pascal

A.1.1. Ejemplo 1

```
program Example;
var
x: integer;
y: integer;
z: longint;
begin
x := 1;
y := 10;
z := 1000000;
x := 20;
writeln(x);
writeln(y);
writeln(z);
end.
```

A.1.2. Ejemplo 2

```
program ExampleProgram;
var
x, y: Integer;
begin
    x := 5;
    y := 10;
    if x > y then
    begin
        writeln(x);
    end
    else
    begin
        writeln(y);
    end;
end.
```

A.1.3. Ejemplo 3

```
program ExampleProgram;
var
    x: Integer;
    i: Integer;
begin
    x := 1;
    for i := 0 to 9 do
    begin
        x := x + i;
    end;
    writeln(x);
end.
```

A.1.4. Ejemplo 4

```
program ExampleProgram;

function suma(a, b: Integer): Integer;
begin
    suma := a + b;
end;
var
    x, y: Integer;
begin
    x := 1;
    y := 20;
    writeln(suma(x, y));
end.
```

A.2. Lenguaje Rust

A.2.1. Ejemplo 1

```
fn main() {
    let mut x: i32;
    let mut y: i32;
    let mut z: i64;
    x = 1;
    y = 10;
    z = 1000000;
    x = 20;
    println!("{}", x);
    println!("{}", y);
    println!("{}", z);
}
```

A.2.2. Ejemplo 2

```rust
fn main() {
    let mut x: i32;
    let mut y: i32;
    x = 5;
    y = 10;
    if x > y
    {
        println!("{}", x);
    }
    else
    {
        println!("{}", y);
    }
}
```

A.2.3. Ejemplo 3

```rust
fn main() {
    let mut x: i32;
    x = 1;
    for i in 0..10
    {
        x += i;
    }
    println!("{}", x);
}
```

A.2.4. Ejemplo 4

```rust
fn suma(a: i32, b: i32) -> i32 {
    a + b
}

fn main() {
    let mut x: i32;
    let mut y: i32;
    x = 1;
    y = 20;
    println!("{}", suma(x, y));
}
```

A.3. Lenguaje Kotlin

A.3.1. Ejemplo 1

```
fun main() {
    var x: Int
    val y: Int
    val z: Long
    x = 1;
    y = 10;
    z = 1000000;
    x = 20;
    println(x)
    println(y)
    println(z)
}
```

A.3.2. Ejemplo 2

```
fun main() {
    var x: Int
    val y: Int
    x = 5;
    y = 10;
    if (x > y) {
        println(x)
    } else {
        println(y)
    }
}
```

A.3.3. Ejemplo 3

```
fun main() {
    var x: Int
    x = 1
    for (i in 0..9) {
        x = x + i
    }
    println(x)
}
```

A.3.4. Ejemplo 4

```
fun suma(a: Int, b: Int): Int {
    return a + b
}
fun main() {
    var x: Int
    var y: Int
    x = 1
    y = 20
    println(suma(x, y))
}
```

A.4. Lenguaje C

A.4.1. Ejemplo 1

```
#include<stdio.h>
int main(){
    int x;
    int y;
    long z;
    x = 1;
    y = 10;
    z = 1000000;
    x = 20;
    printf("%d\n", x);
    printf("%d\n", y);
    printf("%ld\n", z);
    return 0;
}
```

A.4.2. Ejemplo 2

```
#include<stdio.h>
int main(){
    int x;
    int y;
    x = 5;
    y = 10;
    if (x > y){
        printf("%d\n", x);
    } else {
        printf("%d\n", y);
    }
    return 0;
}
```

A.4.3. Ejemplo 3

```c
#include<stdio.h>
int main(){
    int x;
    x = 1;
    for (int i = 0; i < 10; i++){
        x = x + i;
    }
    printf("%d\n", x);
    return 0;
}
```

A.4.4. Ejemplo 4

```c
#include<stdio.h>
int suma(int a, int b){
    return a + b;
}
int main(){
    int x;
    int y;
    x = 1;
    y = 20;
    printf("%d\n", suma(x, y));
    return 0;
}
```

B. Grupos

B.1. Sección 1

Nº	Lenguaje y temas	Integrantes	Exposición
1	KOTLIN, UNSING INT, LONG INT	Izaguirre Zavaleta, Luis Fernando Wong Orrillo, Jose Francisco Sobenes Obregon, Carlos Sebastian	Martes 01-07
2	RUST, ARRAY, STRING	Paca Sotero, Jose Francisco Lezama Orihuela, Sergio Sebastian Gonzalez Vidalon, Ian Steve	Martes 01-07
3	KOTLIN, STRUCT, PUNTEROS	Niño Castañeda, Jesus Valentin Hinojosa Bittrich, Michael Paul	Martes 01-07
4	C, UNSING INT, LONG INT	Cordova Flores, Salvador Justo Ruben Usurin Arias, Fernando Alonso	Martes 01-07
5	PASCAL, ARRAYS, PUNTEROS	Felix Aponte, Renzo Josimar Cueva Mendoza, Manyory Estefany Canto Vidal, Harold Alexis Victor	Jueves 03-07
6	PASCAL, LONGINT, UNSIGN INT	Arias Romero, Jose Armando Aguilar Millones, Jose Ignacio Inca Acuña, Juan Rodolfo	Jueves 03-07
7	RUST, STRUCT, LONGINT	Nagamine Oshiro, Laura Gabriela Bracamonte Toguchi, Mikel Dan Aragon Ayala, Eduardo Fernando	Jueves 03-07
8	PASCAL, STRUCT, FLOAT	Coorahua Peña, Ruben Aaron Quicaña Erquinio, Jefersson Kevin Velarde Tipte, Jesus Gadiel	Jueves 03-07
9	KOTLIN, FLOAT, ARRAYS	Condor Blas, Ronal Jesus Raza Estrada, Gilver Alexis Salinas Salas, Joaquin Mauricio	Sábado 05-07
10	¿?	Chamochumbi Gutierrez, Alexandro Condori Palomino, Jose Eduardo Echarre Lopez, Claudio Huamani Ñaupas, Jose Eduardo	Sábado 05-07

B.2. Sección 2

.

Nº	Lenguaje y temas	Integrantes	Exposición
1	C, STRUCT, PUNTEROS	Lizardo Mejia, Zamir Rogger Meneses Roncal, Matías Alonso	Martes 01-07
2	RUST, FLOAT, ARRAYS	Huaylla Huillca, Jean Piero Arteaga Alvarez, Gian Marco	Martes 01-07
3	C, STRING, ARRAYS	Salazar Mendoza, Sofia del Pilar Salcedo Quiroz, Ariann Abigail Hilario Quintana, Jeffry Arturo	Martes 01-07
4	KOTLIN, STRUCT, ARRAYS	Guerrero Jimenez, Piero Jesus Angeles Barazorda, Jean Pier Renzo Diaz Hurtado, Vasco	Jueves 03-07
5	C, UNSING, LONG	Melgarejo Castillo, Jorge Eduardo Nieto Paz, Hector Sebastian	Jueves 03-07
6	KOTLIN, STRING, FLOAT	Simeón Sarmiento, Isaac Emanuel Javier Galvez Pacori, Jose Guillermo	Jueves 03-07
7	PASCAL, ARRAYS, PUNTEROS	Murakami Miyahira, Mitsuo Sebastian Escajadillo Guerrero, Diego Antonio Flores Teniente, Enrique Francisco	Sábado 05-07
8	¿?	Condori Flores, Ian Kevin Enciso Lozano, Diego Sebastian Guillen Rodriguez, Fernando Andre	Sábado 05-07

C. Salidas

C.1. Ejemplo 1

```
print(1+2);
print(3+4)
```

C.1.1. Salida 1

```
.data
print_fmt: .string "%ld\n"
.text
.globl main
main:
pushq %rbp
movq %rsp, %rbp
movq $1, %rax
pushq %rax
movq $2, %rax
movq %rax, %rcx
popq %rax
addq %rcx, %rax
movq %rax, %rsi
leaq print_fmt(%rip), %rdi
movl $0, %eax
call printf@PLT
movq $3, %rax
pushq %rax
movq $4, %rax
movq %rax, %rcx
popq %rax
addq %rcx, %rax
movq %rax, %rsi
leaq print_fmt(%rip), %rdi
movl $0, %eax
call printf@PLT
movl $0, %eax
leave
ret
.section .note.GNU-stack,"",@progbits
```

C.2. Ejemplo 2

```
var int x,y;
x = 5;
y = 4;
print(x+y)
```

C.2.1. Salida 2

```
.data
print_fmt: .string "%ld\n"
.text
.globl main
main:
pushq %rbp
movq %rsp, %rbp
subq $16, %rsp
movq $5, %rax
movq %rax, -8(%rbp)
movq $4, %rax
movq %rax, -16(%rbp)
movq -8(%rbp), %rax
pushq %rax
movq -16(%rbp), %rax
movq %rax, %rcx
popq %rax
addq %rcx, %rax
movq %rax, %rsi
leaq print_fmt(%rip), %rdi
movl $0, %eax
call printf@PLT
movl $0, %eax
leave
ret
.section .note.GNU-stack,"",@progbits
```

C.3. Ejemplo 3

```
var int x,y;
x = 5;
y = 1;
while 0<x do
y = x*y;
x = x-1
endwhile;
print(y)
```

C.3.1. Salida 3

```
.data
print_fmt: .string "%ld\n"
.text
.globl main
main:
pushq %rbp
movq %rsp, %rbp
subq $16, %rsp
movq $5, %rax
movq %rax, -8(%rbp)
movq $1, %rax
movq %rax, -16(%rbp)
while_0:
movq $0, %rax
pushq %rax
movq -8(%rbp), %rax
movq %rax, %rcx
popq %rax
cmpq %rcx, %rax
movl $0, %eax
setl %al
testq %rax, %rax
je endwhile_1
subq $16, %rsp
movq -8(%rbp), %rax
pushq %rax
movq -16(%rbp), %rax
movq %rax, %rcx
popq %rax
imulq %rcx, %rax
movq %rax, -16(%rbp)
movq -8(%rbp), %rax
pushq %rax
movq $1, %rax
movq %rax, %rcx
popq %rax
subq %rcx, %rax
movq %rax, -8(%rbp)
jmp while_0
endwhile_1:
movq -16(%rbp), %rax
movq %rax, %rsi
leaq print_fmt(%rip), %rdi
movl $0, %eax
call printf@PLT
movl $0, %eax
leave
ret
.section .note.GNU-stack,"",@progbits
```