2do Examen parcial de Compiladores 12 mayo

Nombre Alumno(a): Guadalupe Sugeily Cruz Cervantes

Nota importante: Textos copiados y pegados de internet no son válidos

Primera parte del examen valor 50%

1.- -¿Qué programas son más rápidos, los interpretados o los compilados?

Los programas compilados son más rápidos que los interpretados.

2.- -Defina compilador cruzado.

Es un compilador que produce código para una plataforma distinta a aquella donde se ejecuta.

3.- -Haga una tabla de diferencias entre un intérprete y un compilador

Programa compilado	**Programa interpretado**
Mas rápido	Mas lento
Variables estáticas	Variables dinámicas
Genera código objeto	No genera código objeto
Solo se puede ejecutar si no tiene errores	Se puede ejecutar hasta encontrar el primer error grave

4.- - ¿Los programas interpretados manejan tipos estáticos o dinámicos?

Tipos Dinámicos

5.- -Explique por qué es más eficiente un INC i; que LDA i; ADD 1; STA i;

INC es una instrucción de conteo, es más eficiente porque lo hace en solo un ciclo de reloj.

6.- -¿Cuál es la principal diferencia entre arquitecturas Harvard y Von Newman?

La diferencia principal entre estas arquitecturas se localiza en el mapa de memoria;

- Von Neumann hay un único espacio de memoria para datos y para instrucciones.
- Harvard hay dos espacios de memoria separados: un espacio de memoria para los datos y un espacio de memoria para las instrucciones.

7.- -Explique en qué consisten la arquitectura Harvard

Es una configuración de la computadora en la que las instrucciones y los datos de un programa se localizan en celdas separadas de memoria, las cuales se pueden abordar de forma independiente. Contiene dos áreas separadas: para los comandos o instrucciones y para los datos.

8.- -Explique en qué consisten la arquitectura Von Newman

Es la arquitectura en la que se basan todos los procesadores para PC, puesto que todos ellos están organizados con una serie de componentes comunes, como lo son la Unidad de Control, Unidad logico-aritmética o ALU, Memoria, Dispositivo de entrada, Dispositivo de Salida. Este tipo de máquina es la implementación de una máquina de Turing y la visión de una arquitectura secuencial en lugar de paralela.

9.- -Explique en qué consiste la filosofía RISC:

L as instrucciones comunes se utilizan eficientemente. Es necesario implementar diferentes funciones combinando instrucciones. Es una filosofía de diseño en código ensamblador de CPU que esta a favor de conjuntos de instrucciones pequeñas.

10.- -¿Cuál es la diferencia entre las arquitecturas RISC y CISC?

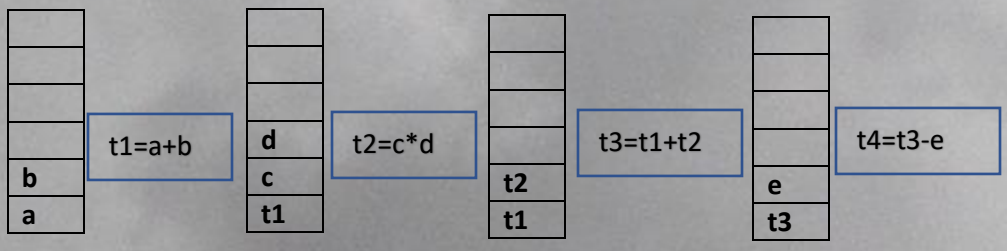
Una de las diferencias entre estos dos diseños son los transistores ya que RISC tiene 35 000 transistores y CISC 2 000 000 000 transistores La arquitectura RISC funciona con conjunto reducido de instrucciones en cuanto a la CISC con conjunto complejo de instrucciones.

RISC	CISC
Unas cuantas instrucciones simples	Muchas instrucciones complejas
Instrucciones de longitud fija	Instrucciones de longitud variable
Complejidad en el compilador	Complejidad en el Microcódigo
Acceso a la memoria solo con instrucciones load/store	Muchas instrucciones pueden acezar la memoria
Muy pocos modos de Direccionamiento	Muchos modos de Direccionamiento

Segunda parte del examen, valor 50%

11.- Convierte la siguiente expresión posfija a codigo intermedio y a código: ab+cd*+e-

Codigo Intermedio

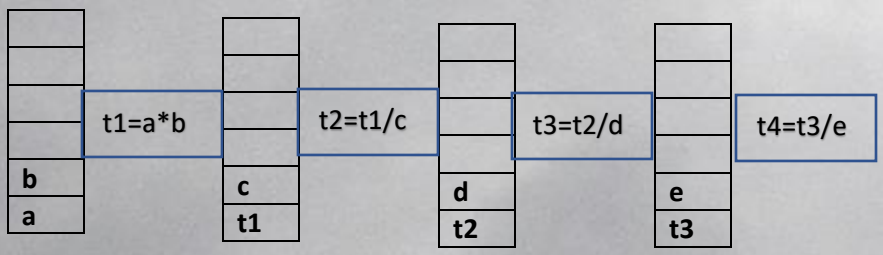


Codigo Ensamblador

LDA a;
ADD b;
STA t1;
LDA c;
MUL d;
STA t2;
LDA t1;
ADD t2;
STA t3;
LDA t3;
SUB e;
STA t4;

12.- Convierte la siguiente expresión posfija a código intermedio y a código: ab*c/d/e/

Código Intermedio

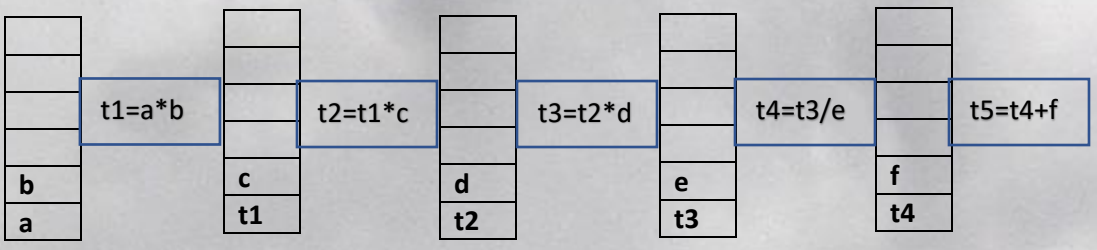


Código Ensamblador

LDA a;
MUL b;
STA t1;
LDA t1;
DIV c;
STA t2;
LDA t2;
DIV d;
STA t3;
LDA t3;
DIV e;
STA t4;

13.- Convierte la siguiente expresión posfija a codigo intermedio a código y optimice en caso de ser posible, (escriba tanto el código sin optimizar como el optimizado): ab*c*d*e/f+

Código Intermedio



Código Ensamblador

```
LDA a;
MUL b;
STA t1;
LDA t1;
MUL c;
STA t2;
LDA t2;
MUL d;
STA t3;
LDA t3;
DIV e;
STA t4;
LDA t4;
ADD f;
STA t5;
```

Código Optimizado

```
LDA a;
MUL b;
MUL c;
MUL d;
DIV e;
ADD f;
STA t5;
```

14.- Convierte la siguiente expresión posfija a codigo intermedio a código y optimice en caso de ser posible, (escriba tanto el código sin optimizar como el optimizado): ab*c-de*-f-

Código Intermedio

```
t1=a*b
t2=t1-c
t3=d*e
t4=t2-t3
t5=t4-f
```

Código Ensamblador

```
LDA a;
MUL b;
STA t1;
LDA t1;
SUB c;
STA t2;
LDA d;
MUL e;
STA t3;
LDA t2;
SUB t3;
STA t4;
LDA t4;
SUB f;
STA t5;
```

Código Intermedio

```
LDA a;
MUL b;
SUB c;
STA t2;
LDA d;
MUL e;
STA t3;
LDA t2;
SUB t3;
SUB f;
STA t5;
```