PRIMER PARCIAL
ARQUITECTURA DE COMPUTADORES

Nombre:	**Fecha: 14 septiembre *2018***
Código:	**Duración: *2 Horas***

No se permite el uso ni la tenencia de ningún elemento electrónico de almacenamiento de datos, Smartphone o computador durante el examen, ni uso de aplicaciones o herramientas de conversión numérica.

1.0 (70 %)

El desarrollo de la inteligencia artificial en la aviación comercial justo empieza a dar los primeros pasos. Gracias a los computadores abordo se está ampliando las capacidades del piloto automático.
Suponga que usted es contratado por una aéreo-línea para iniciar las primeras pruebas para el uso del piloto automático en el despegue de un avión, tarea que hasta ahora se hacía siempre de forma manual.

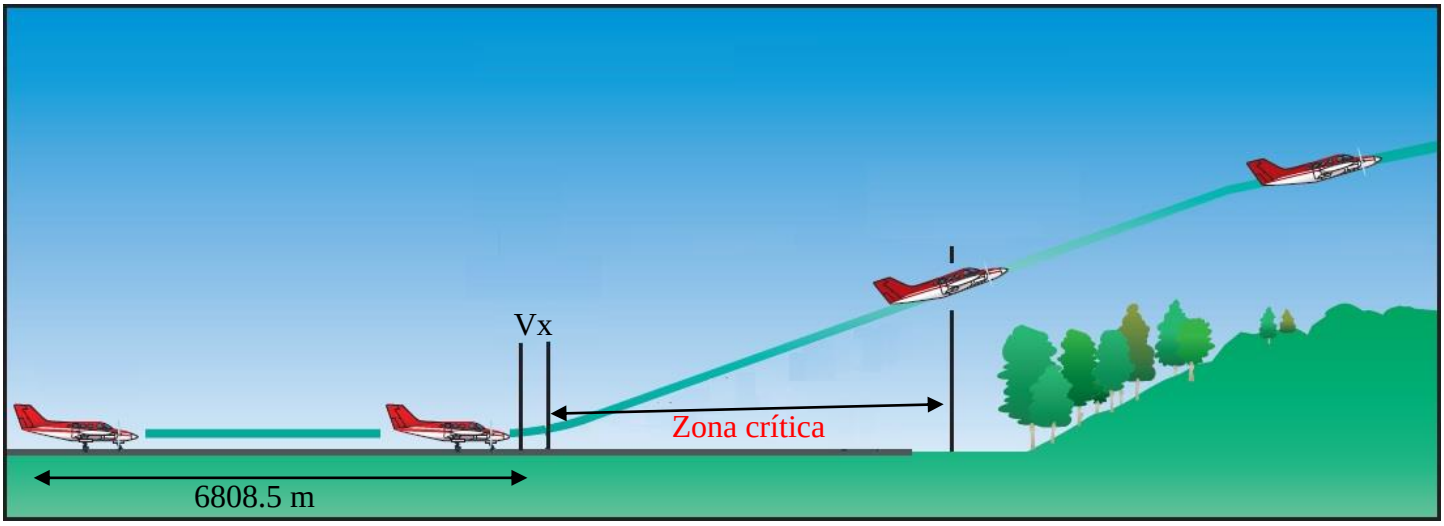

Figura 1

La física mecánica del despegue es muy clara, y el piloto automático ahora es quién debe tomar la decisión de levantar vuelo solo si se ha alcanzado la velocidad de despegue mínima Vx de 332.1 km/h. Por encima de dicha velocidad el avión podrá elevarse. Un intento de despegue a una velocidad inferior a Vx terminaría en choque abrupto de la trompa del avión contra el suelo.
Ahora si existen obstáculos al final de la pista (como se ve en la figura 1), entonces también se debe considerar iniciar el despegue solo si se encuentra por fuera de la zona crítica de distancia horizontal al obstáculo, si se inicia el ascenso estando muy cerca al obstáculo, el avión no alcanza a sobre volarlo y se estrellará.
Suponiendo que el avión parte del reposo ($v_0 = 0$) con aceleración constante a = 0.625 m/s^2 y que la velocidad de despegue Vx debe ser 332.1 km/h.

Se aplica entonces las ecuaciones de movimiento y se realizan las conversiones de unidades como sigue:

La velocidad, expresada en unidades SI, es: $332.1\frac{km}{h}.\frac{1000\,m}{1\,km}.\frac{1h}{3600\,s} = \mathbf{92.25}\frac{\boldsymbol{m}}{\boldsymbol{s}}$

Usamos la ecuación $\boldsymbol{v_f^2 = v_0^2 + 2ad}$ y despejamos el valor de la distancia *d* donde el avión alcanza la velocidad de despegue:

$$d = \frac{{V_f}^2}{2a} \approx \mathbf{6808.05}\,\boldsymbol{m}$$

Para determinar el tiempo en que se alcanza dicha velocidad de despegue usamos la ecuación $v = v_0 + at$ y despejamos el tiempo:

$$t = \frac{v}{a} = \frac{92.25\frac{m}{s}}{0.625\frac{m}{s^2}} \approx 147.6s$$

Entendiendo las variables y condiciones del problema, entonces suponga que un ingeniero aeronáutico escribe el siguiente programa de JAVA:

Figura 1

```
1  public class MyClass {
2      public static void main(String args[]) {
3          float _16 DeltaTime=0.09375;
4          float _16 time=0.0f;
5          float _16 Vx=92.25f; //velocidad de despegue
6          float _16 v=0; // velox actual
7          float _16 a=0.625f; //Aceleración constante
8          float _16 x=0.0f; //distancia recorrida
9          float _16 xLimite=6808.05f; //Distancía límite segura
10         int _16   i =0;
11
12         while (x<=xLimite){
13             time=time+DeltaTime;
14             v=(a*time);
15             x=((v*v))/(2*a);
16             i++;
17             if(v>=Vx){
18              System.out.println("Despegue");
19              break;
20             }
21         }
22
23     }
24 }
```

Como el computador abordo trabaja con una CPU de 16 bits, entonces las variables son todas de 16 bits. Entonces la variable **float_16** está en formato IEEE 754 punto flotante de 16 bits, donde 5 bits son para el exponente y 10 para la mantisa. **Int16** hace referencia a enteros de 16 bits.

El programa se basa en una estructura while, donde se itera mientras la distancia recorrida por el avión x sea menor o igual 6808.05 m;

En cada iteración se incrementa una variable `time` que refleja un reloj interno del sistema que se incrementa en **DeltaTime**=`0.09375s.` Como se supone que el avión parte del reposo y avanza siempre con una aceleración constante 0.625 m/s2, entonces el programa va calculando *x* y *v* usando el valor de la variable *time*.

Como los ingenieros de sistemas de la Universidad Icesi son reconocidos por su capacidad de análisis, entonces la aerolínea le pide a usted que analice el programa anterios y detecte posibles problemas relacionados con la aritmética del procesador, se pide entonces lo siguiente

A. Indique la representación de los valores iniciales de las siguientes variables: (12%)

Variable	Representación binaria de 16 bits	hexadecimal
DeltaTime		
Vx		
xLimite		

B. Analice el código y describa todas las fuentes posibles de error que se pudieron dar por la aritmética binaria que maneja el computador, incluyendo el cálculo de la variable *time.* Describa claramente cada uno de los errores detectados. (10%)

C. Si se pone en ejecución del programa y en marcha el avión, indique que distancia que ha recorrido el avión desde su punto de reposo hasta que alcanza una velocidad mayor o igual a Vx. Y entonces responda si se logra un despegue normal, o por si el contrario el avión intenta despegar antes o después de la distancia límite. (20%)

D. En el caso de que haya detectado problemas en los formatos de numeración usado o en la estructura del programa, entonces proponga maneras para corregir los errores y de esa manera minimizar o eliminar los efectos de las fuentes de error. Si cree que el problema está en el tipo de variable float_16, entonces proponga un nuevo formato de punto flotante donde la cantidad de bits sea apenas la necesaria para cumplir con las condiciones del problema, es decir determine la cantidad mínima de bits que se necesitaría para el exponente y para la mantisa. Calcule para el nuevo formato el rango y la mejor precisión (18%)

E. Si se cambia el deltaTime de `0.09375s a 0.093s` cuál sería la nueva distancia recorrido del avión antes del despegue, ¿habría alguna diferencia¡. (10%).

3.0 (30%)

Dado el esquema de bloques de la arquitectura de un procesador de 16 bits de la figura 2:

Considere el conjunto de instrucciones –escrito en lenguaje de alto nivel- del siguiente programa, donde se operan variables tipo **short** (enteros de 16 bits). Suponga que se desea ejecutar el programa usando el computador de la figura 1, que se basa a su vez en un procesador de 16 bits.

```
 9  short sA,sB;
10  unsigned short uA;
11
12  int main()
13  {
14      sA=6FF8h;
15      sB=40CDh;
16      sB=sB*2;
17      uA=sB-sA;
18  }
```

Lenguaje Ensamblador	Lenguaje máquina	

a) Escriba sobre la tabla, el programa en lenguaje ensamblador que sea equivalente al programa anterior usando el ISA del procesador de la figura 1 (Asuma que en lugar de posiciones de memoria se usan registros para almacenar los valores de sA, sB y uA). Como es un procesador RISC muy básico, asuma que el repertorio de instrucciones se reduce a las cinco instrucciones descritas en el anexo. Escriba al frente de cada instrucción ensamblador el equivalente al lenguaje máquina y expréselo además en hexadecimal. (15%)

b) Muestre además el estado de las banderas de estado (Z, S, O y C) y el contenido de cada registro(s) involucrado(s) después de la ejecución de cada una de las instrucciones en ensamblador correspondientes a las líneas 16 y 17. (10%)

c) Explique qué operaciones y que banderas debe considerar la CPU para comparar números enteros con signo y números enteros sin signo (10%)

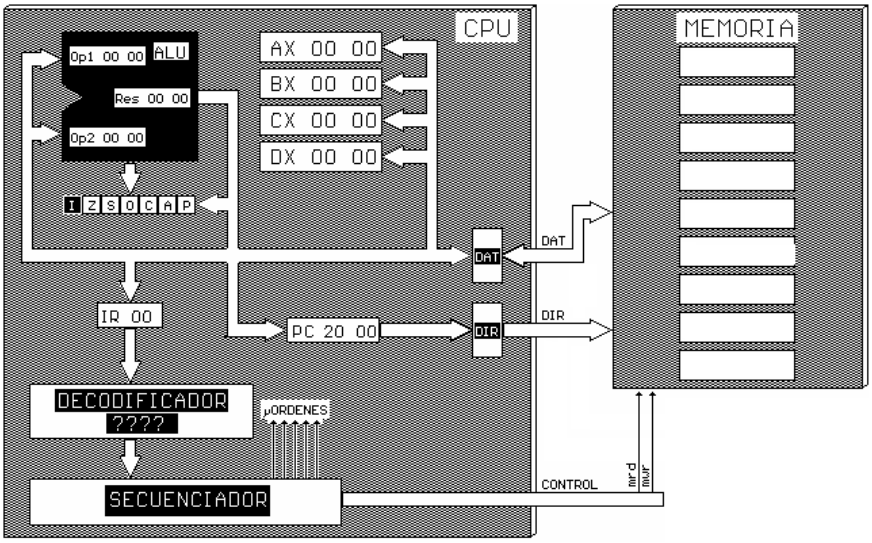

Figura 2

ANEXO

MOV registro, dato inmediato

1011Wrrr	LSB dato	MSB dato

SUB registro, dato inmediato

1000000W	11101rrr	LSB dato	MSB dato

ADD registro, REGISTRO

0000001W	11rrrRRR

MOV registro, REGISTRO

1000101W	11rrrRRR

SUB registro, REGISTRO

0010101W	11rrrRRR

W	SIGNIFICADO
0	operandos de 8 bits
1	operandos de 16 bits

r R	r R	r R	W = 0	W = 1
0	0	0	AL	AX
0	0	1	CL	CX
0	1	0	DL	DX
0	1	1	BL	BX
1	0	0	AH	SP
1	0	1	CH	- -
1	1	0	DH	- -
1	1	1	BH	--

- **rrr, RRR:** Referencian a los registros (**registro, REGISTRO,** respectivamente) implicados en la instrucción, de acuerdo a la siguiente tabla: