

UTN - INSPT
2.601 - Sistemas de Computación I
Parcial 2021

Apellido(s), Nombre(s): Alonso Lucio

Plantee y resuelva las siguientes cuestiones. Deberá insertar debajo de cada una de las preguntas 1 a 9, claramente legibles de su puño y letra, los pasos seguidos detalladamente y la solución (pueden ser imágenes escaneadas o fotos). La explicación del punto 10 deberá ser de puño y letra y la implementación deberá estar subida en GitLab o GitHub, con el docente invitado a verla como reporter.

El parcial resuelto deberá enviarse, en un archivo PDF, hasta el 09/11/2021 a: corsi@mail.com

1) ¿Cuál es, en decimal, la representación del número octal -127,243?

The image shows a handwritten solution on a grid background. It starts with the question: ① $(-127,243)_8 \text{ a } 10.$. Below this, the octal number is expanded into its decimal equivalent using powers of 8. The digits 1, 2, 7, 2, 4, and 3 are positioned above the terms 8^2 , 8^1 , 8^0 , 8^{-1} , 8^{-2} , and 8^{-3} respectively, with arrows pointing down. The expression is:
$$-(1 \times 8^2) + (2 \times 8^1) + (7 \times 8^0) + (2 \times 8^{-1}) + (4 \times 8^{-2}) + (3 \times 8^{-3}) = -87,318359375$$
 The final result, -87,318359375, is enclosed in a rectangular box.

- 2) ¿Cuáles son, al usar Complemento a 2 (C-2), los 16 bits de la representación del número entero -24683?

② Primero paso el número de decimal a binario

Handwritten conversion of 24683 to binary using repeated division by 2:

24683	12
12341	2
6170	2
3085	2
1542	2
771	2
385	2
192	2
96	2
48	2
24	2
12	2
6	2
3	2
1	2

Resulting binary sequence (from bottom to top): 110000001101011

Handwritten conversion of 12 to binary:

12	2
6	2
3	2
1	2

Resulting binary sequence (from bottom to top): 1100

Final binary representation: $(24683)_{10} = 110000001101011$

Segundo agrego la cantidad de 0 hasta llegar a 16 dígitos

0110000001101011

Tercero aplico complemento a 2-1 (cambio los 0 por 1 y los 1 por 0).

1001111110010100

Por último al C-1 le sumamos 1.

Final result: 1001111110010101

- 3) ¿Cuál es, al usar IEEE 754 (single), el número decimal representado por los 32 bits expresados mediante los 8 dígitos hexadecimales ABAD2020?

③ Primero paso el número Hexadecimal a binario

$(ABAD2020)_{16} = (10101011101011010010000001000000)_2$

Segundo: Separa el primer bit, y que representa el signo, los siguientes 8 bits que representan el exponente, el resto de los 23 bits restantes, son la mantisa.

Para todos los conjuntos a decimal. Luego realizo el cálculo.

S exponente Mantisa

1 0 1 0 1 0 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0

$S = 1$

$V = (-1)^S \cdot 2^{E-127} \cdot 1, M$

$Exp = 01010111 \rightarrow 1+2+4+16+64 = 87 - 127 = -40$

$Mant = 01011010010000000100000 \rightarrow 1,010110100100000001$

$V = (-1)^1 \cdot 2^{-40} \cdot 1,352542877 = 1,230130581 \cdot 10^{-12}$

$(2^{-2} + 2^{-4} + 2^{-5} + 2^{-7} + 2^{-10} + 2^{-18}) = 1,352542877$

- 4) ¿Cuál es el número que, al ser representado usando el Código 8421 (BCD natural), da como resultado la secuencia binaria 001101101000?

④ BCD Natural

$001101101000 = 368_{10}$

Primero separo la secuencia de 4.

Luego paso cada grupo de 4 dígitos a decimal.

Junto los números y con esto conseguí el resultado.

- 5) ¿Cuál es, al usar el Código ISO/IEC 8859-1, la cadena representada por los 56 bits 00110101 11010001 01110010 01000101 01100001 00100000 00110001?

5) La cadena representada por los 56 bits es $\tilde{S}\tilde{N}rEa1$

Para obtenerlo, busque la tabla donde se encuentra el valor decimal conjunto de bits.

Luego pase los datos a hexadecimal y busque en la tabla el carácter que equivale.

$$\begin{array}{c} \text{Nro. Bin} \\ \hline 00110101 \\ \hline 3 \quad 5 \end{array} = 35 = 5$$

$$\begin{array}{c} \hline 11010001 \\ \hline D \quad 1 \end{array} = D1 = \tilde{N}$$

$$\begin{array}{c} \hline 01110010 \\ \hline 7 \quad 2 \end{array} = 72 = r$$

$$\begin{array}{c} \hline 01000101 \\ \hline 4 \quad 5 \end{array} = 45 = E$$

$$\begin{array}{c} \hline 01100101 \\ \hline 6 \quad 1 \end{array} = 61 = a$$

$$\begin{array}{c} \hline 00100000 \\ \hline 2 \quad 0 \end{array} = 20 = \text{Espacio}$$

$$\begin{array}{c} \hline 00110001 \\ \hline 3 \quad 1 \end{array} = 31 = 1$$

- 6) ¿Cuáles son los 8 bits con los que se representa el número decimal 185 al usar el código binario reflejado de Gray?

⑥ Primero paso el 185 decimal a binario

$$\begin{array}{r}
 185 \div 2 \\
 \hline
 18 \quad 92 \div 2 \\
 \hline
 05 \quad 46 \div 2 \\
 \hline
 4 \quad 23 \div 2 \\
 \hline
 1 \quad 11 \div 2 \\
 \hline
 0 \quad 5 \div 2 \\
 \hline
 0 \quad 2 \div 2 \\
 \hline
 1 \quad 1 \div 2 \\
 \hline
 0
 \end{array}$$

$$(185)_{10} = (10111001)_2$$

Luego paso el número binario a código de Gray

Mantengo el bit más significativo y luego de izquierda a derecha hago una suma binaria de bits y descarto el acarreo.

$$(10111001)_2 = \boxed{(11100101)_{\text{Gray}}}$$

$$\begin{array}{cccccccc}
 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\
 \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1
 \end{array}$$

- 7) ¿Cuáles fueron los 7 bits enviados usando el código de Hamming (7, 4), si se recibió la siguiente secuencia binaria con (como máximo) un bit erróneo? 1010110

⑦ Primero marco los bits de paridad y de datos

b ₁	b ₂	b ₃	b ₄	b ₅	b ₆	b ₇
1	0	1	0	1	1	0
p ₁	p ₂	d ₁	p ₃	d ₂	d ₃	d ₄

Verifico la paridad

se necesita un 0

$$p_1 = \text{paridad}(d_1, d_2, d_4) \quad p_1 = 0, d_1 = 1, d_2 = 1, d_4 = 0$$

$$p_2 = \text{paridad}(d_1, d_3, d_4) \quad p_2 = 0, d_1 = 1, d_3 = 1, d_4 = 0$$

$$p_3 = \text{paridad}(d_2, d_3, d_4) \quad p_3 = 0, d_2 = 1, d_3 = 1, d_4 = 0$$

Al analizar, encontramos que en el bit 1 no hay paridad y que el error se encuentra ahí. (Aplicando compuertas XOR se llega a la misma conclusión.)

Corrigiendo, el código final quedará: 0010110

- 8) ¿Cuántos accesos a RAM se requieren para ejecutar la instrucción POP EBX? Explique y justifique su respuesta

⑧ Se requieren dos accesos a la memoria ram. El primero para desapilar el dato y el segundo para almacenarlo en EBX.

- 9) ¿Qué modo de direccionamiento utiliza la instrucción MOV BX, [4691]? Explíquelo basándose en la codificación de la instrucción cuando está alojada en la RAM.

⑨ El modo de direccionamiento que emplea la instrucción MOV BX, [4691] es el directo. Lo que hace es copiar en BX los 16 bits a partir 4691.

- 10) Explique e implemente las modificaciones requeridas por su compilador de PL/0 para que, a partir del código fuente que se muestra a continuación, genere un archivo PE-32 (para Windows) o ELF (para GNU/Linux) que corra según los ejemplos.

```
const MIN = -10000, max=10000;

var A,B,C,W,X,Y,Z;

procedure MOSTRAR;
begin
  IF Y = 0 THEN WRITELN ('HAY UNA RAIZ EN X=', X);
  W := 1
end;

begin
  WRITELN;
  WRITELN ('PROGRAMA PARA BUSCAR RAICES ENTERAS DE ECUACIONES CUADRATICAS');
  WRITELN;
  WRITELN ('  2');
  WRITELN ('A X + B X + C = 0');
  WRITELN;
  write ('Ingrese A: '); readLn (A);
  write ('Ingrese B: '); readLn (b);
  write ('Ingrese C: '); readLn (c);
  WRITELN;
  x := min;
  W := 0;
  while x <= max do
    begin
      z := x - 1;
      y := a * sqr(z+1) + B * x + c;
      if y = 0 then call MOSTRAR;
      x := x + 1
    end;
  if W = 0 then writeln ('NO HAY RAICES ENTERAS ENTRE ', MIN, ' Y ', MAX)
end.
```

Ejemplos:

$$2x^2 - 6x + 4 = 0$$

Tiene 2 raíces enteras ($x = 1$; $x = 2$)

$$3x^2 = 0$$

Tiene una raíz entera ($x = 0$)

$$x^2 - 1 = 0$$

Tiene 2 raíces enteras ($x = -1$; $x = 1$)

$$2x^2 + 2x + 2 = 0$$

No tiene raíces enteras

10) Las modificaciones implementadas en el compilador fueron las siguientes:

1° Se agregó en la función BLOQUE (dentro del if que detecta un símbolo CONST), que luego de leer un IGUAL (=), pueda leerse a continuación un NEGATIVO (-) y un número. Si no entra un negativo, se busca un número.

```
Código: if ((*simbolo) == - MENOS)
{
    cdenz = simboloNeg;
    strcpy (simboloNeg, cdenzSimbolo);
    escner (f, restante, simbolo, cdenzSimbolo, contadorPanglon);
    if ((*simbolo) == NÚMERO)
    {
        strcpy (simboloNeg, cdenzSimbolo);
        strcpy (cdenzSimbolo, simboloNeg);
        tblzSimb [base + desplazamiento].valor = atoi (cdenzSimbolo);
        escner (f, restante, simbolo, cdenzSimbolo, contadorPanglon);
    }
    else mensajeErr (f, s, cdenzSimbolo);
}
```

2° Luego se modificó la función búsqueda, ya que esta diferenciar entre minúsculas y mayúsculas. Esto provocaba que a la hora de querer leer una variable que había sido escrita con mayúsculas en su declaración, no se la logre reconocer como tal si más adelante en el código se la escribía en minúsculas.

Para arreglar esto, se transformó, en la comparación de los símbolos, que ambos estén en mayúsculas usandostrup.

Código: Ej: `if (strcmp(strupr(nombreSimb), strupr(tblSimb[i].nombre)) == 0) { break; }`

3° Por último, se agregó el SQR (para evaluar la potencia) al código.

- Primero se colocó dentro del typedef enum el SQR.
- Segundo, dentro de la función factor, se agregó un caso SQR, el cual es el encargado de evaluar la potencia. Detecta la apertura de un paréntesis, espera una expresión, luego hace los pushes de memoria correspondientes para evaluar la potencia del número y por último, espera un cierre de paréntesis.

Código: caso SQR:

```
escanear(f, restante, simbolo, cdenzSimb, contadorRenglon);
if ((*simbolo) == ABREPARENTESIS) {
    escanear(f, restante, ...);
    expresion(f, restante, simbolo, cdenzSimb, contadorRenglon);
    tblSimb[bzse, desplazamiento, topeMemoria, memoria);
    cargarByte(0x5B, memoria, topeMemoria); // POP EBX
    cargarByte(0xB8, memoria, topeMemoria); // MOV EAX, X
    cargarInt(0x00, memoria, topeMemoria);
    cargarByte(    , memoria, topeMemoria); // ADD EAX, EBX
    cargarByte(    , memoria, topeMemoria);
    cargarByte(    , memoria, topeMemoria); // IMUL EBX
    cargarByte(    , memoria, topeMemoria);
    cargarByte(    , memoria, topeMemoria); // push EAX
    if ((*simbolo) == CIERRAPARENTESIS) {
        escanear(...);
    } else...
```

- En la función `se agregó el símbolo SQR.`

Código: `else if (strcmp(cadenaAux, "SQR") == 0)`
`{ *simbolo = SQR; }`

- Y en la función `imprimirSimbolo` se agregó un `case SQR`

Código: `case SQR:`
`printf("SQR\n");`
`break;`