# Examen de Arquitectura de Computadoras

# 22 de febrero de 2023

**Instrucciones:**

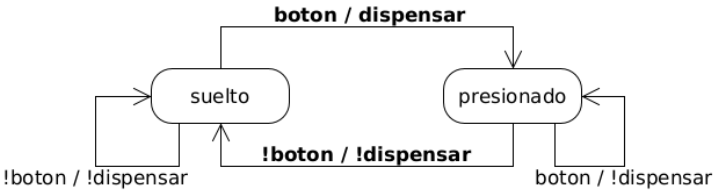
- **Indique su nombre, apellido y número de cédula en todas las hojas que entregue.**
- **Escriba las hojas de un solo lado. Empiece cada ejercicio en una hoja nueva.**
- **Las hojas deben estar numeradas y en la primer hoja debe escribirse el total.**
- **Apague su celular. No puede utilizar material ni calculadora.**
- **La duración del examen es de tres horas. En dicho tiempo debe también completar sus datos.**
- **Para aprobar debe contestar correctamente un ejercicio entero y dos preguntas.**

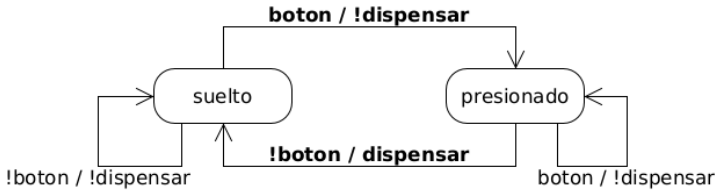
Pregunta 1

a) Calcular el mínimo entero *i* tal que $2^i$ es representable como número **normalizado** en punto flotante IEEE 754 de simple precisión. Justifique.
b) Calcular el mínimo entero *i* tal que $2^i$ es representable como número **desnormalizado** en punto flotante IEEE 754 de simple precisión. Justifique.

### *Pregunta 2*

Las siguientes máquinas de estados modelan un sistema dispensador de caramelos. El sistema tiene como entrada un botón que bebe presionar el usuario para dispensar caramelos, y genera una salida que indica que deben liberarse caramelos.




a) Desde el punto de vista del usuario, ¿cuál es la diferencia de funcionamiento de las máquinas?
b) Siguiendo la metodología del curso, diseñe el circuito que implementa la máquina de la izquierda, utilizando flip-flops D y compuertas lógicas.

### *Pregunta 3*

¿Cuál es la diferencia entre los modos de direccionamiento directo e indirecto? ¿Cuántos accesos a memoria se requieren en cada tipo de direccionamiento y sus variantes para obtener un operando?

### *Pregunta 4*

Describa el ciclo de instrucción. Indique las acciones desarrolladas por la Unidad de Control en la etapa de Read.

## *Problema 1*

**a)** Implemente un algoritmo recursivo que determina si un string es palíndrome. El string está almacenado en un arreglo. A continuación se presenta el encabezado de la función:

short esPalindrome(char *str, short izq, short der);

Inicialmente es invocada de la siguiente manera: esPalindrome(str, 0, len(str)-1).

Compilar la función esPalindrome a asembler 8086. Los punteros son near relativos a DS. Los parámetros y el resultado deben pasarse por el stack.

**b)** Determinar el tamaño máximo del string para que la función pueda ejecutarse correctamente en cualquier caso sabiendo que el comienzo del string es 0x9000.

**Nota**: palíndrome: palabra o frase cuyas letras están dispuestas de tal manera que resulta la misma leída de izquierda a derecha que de derecha a izquierda [RAE].

## *Problema 2*

La empresa ***Taquenchi*** les ha encargado la programación del control principal de su nuevo *climatizador* de piscinas ("bomba de calor"). Un *climatizador* utiliza un motor compresor para tomar calor del ambiente y traspasarlo al agua que circula por él, movida por una bomba externa al dispositivo.
El *climatizador* dispone de una CPU dedicada y los siguientes elementos:

- un motor compresor que se controla a través del bit más significativo del byte de E/S, de solo escritura, accesible en la dirección ***COMPRESOR*** (0 = compresor apagado, 1 = encendido).

- un sensor de la temperatura del agua que circula por él, cuya lectura (en grados Celcius) está accesible en el byte de E/S en la dirección ***TEMP_AGUA***.

- un caudalímetro que permite saber, en todo momento, cuantos decilitros de agua han circulado por él en el último segundo, valor que se puede leer en el byte de E/S en la dirección ***CAUDAL***.

- un display que muestra el valor que se escribe en los 7 bits menos significativos (b6..b0) del byte de E/S, de solo escritura, en la diurección ***DISPLAY***. También enciende un LED que indica alarma si se escribe un 1 en el bit mas significativo de dicho puerto.

- un timer que genera una interrupción con una frecuencia de 1 Hz invocando a la rutina ***timer()***.

A su vez el equipo que desarrolla el control remoto que utiliza el usuario para configurar el *climatizador* deja la información de configuración accesible en los siguientes puertos, al mismo tiempo que genera una interrupción que invoca a la rutina ***configurar()***, para indicar que se configuraron valores nuevos:

- en el byte de E/S en la dirección ***TEMPERATURAS***, en los 6 bits menos significativos (b5..b0) la temperatura ***deseada*** por el usuario para el agua de su piscina y en los dos bits más signitficativos (b7 y b6) el ***delta*** de temperatura que determinará que el compresor debe arrancar de nuevo luego que se alcanzó la temperatura deseada y el compresor fue por tanto apagado.

- en el byte de E/S en la dirección ***ENCENDIDO***, en el bit menos significativo (b0) indica si el usuario ha encendido/apagado el *climatizador* (0 = apagado, 1 = encendido). En el bit b1 indica si ha activado el temporizador de apagado automático, en cuyo caso (b1 = 1) se deberá apagar el compresor luego de tantos minutos de activado el temporizador como indiquen los 6 bits más significativos (b7..b2).

**Se pide:**

Escribir en un lenguaje de alto nivel, preferentemente C, todas las rutinas necesarias para controlar el funcionamiento del compresor, de modo que el *climatizador* tenga un comportamiento determinado por las siguientes condiciones, en tanto el usuario lo mantenga encendido:

- El compresor deberá estar encendido hasta que el agua que circula por él alcance la tempreatura ***deseada***. En ese momento deberá apagarse hasta que dicha temperatura baje de (***deseada – delta)***, siendo ***deseada*** y ***delta*** los valores configurados por el usuario.

- El compresor deberá apagarse en caso que la cantidad de agua que circuló por el *climatizador* en el último minuto sea menor a ***CAUDAL_MINIMO*** litros. Esta verificación debe realizarse una vez por segundo.

- En caso que el usuario active el temporizador de apagado, el compresor deberá apagarse al pasar la cantidad de minutos configurada, contados desde el momento en que dicho temporizador fue activado por el usuario, sin importar si se alcanzó o no la temperatura deseada.

- En todo momento deberá mostrar en el display la temperatura del agua circulante y en caso que se haya tenido que apagar el compresor por bajo caudal deberá, además, encender el LED de alarma.

**Solución**

## *Pregunta 1*

a) El número positivo más pequeño normalizado tiene la siguiente representación:
00000000100000000000000000000000 de donde s=0, E=00000001 (8 bits),
m=00000000000000000000000 (23 bits)

Dicho número es de la forma 1,0000…0 x $2^e$ por lo que el i buscado será igual al menor exponente e.
Como el exponente e se representa en desplazamiento, se sabe que E = e+d, donde
$d=2^{n-1}-1 = 128-1=127$ (n=el número de bits para representar el exponente)

Entonces E=e+d => 1=e+127 => e=-126 => i=-126

b) El número positivo más pequeño desnormalizado tiene la representación:
00000000000000000000000000000001 de donde s=0, E=00000000 (8 bits),
m=00000000000000000000001 (23 bits)

Dicho número desnormalizado es 0,00..01 $x2^{-126}$ (-126 es el exponente por definición de los desnormalizados), por lo que llevado a la forma $2^i$ obtenemos $2^{-126-n}=2^{-126-23}$ (n es el número de bits para representar la mantisa) por lo tanto i=-149

## *Pregunta 2*

a) La diferencia entre las máquinas es que la primera dispensa los caramelos al apretar el botón y la segunda dispensa los caramelos al soltar el botón.

b)

1. Tabla de estados

| Estado actual | boton | Proximo estado | dispensar |
|---|---|---|---|
| suelto | 0 | suelto | 0 |
| suelto | 1 | presionado | 1 |
| presionado | 1 | presionado | 0 |
| presionado | 0 | suelto | 0 |

2 y 3. Cantidad de bits de entrada y salida, cantidad de FF y codificacion:
2 estados → 1 bit
1 FF → salida q
suelto → 0, presionado → 1

4. Tabla de transiciones y salidas

| q | boton | q | dispensar |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |

5. Tabla de verdad

Como el FF es tipo D, $Q_{n+1} = D_n$

| q | boton | d | dispensar |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |

6. Karnaugh

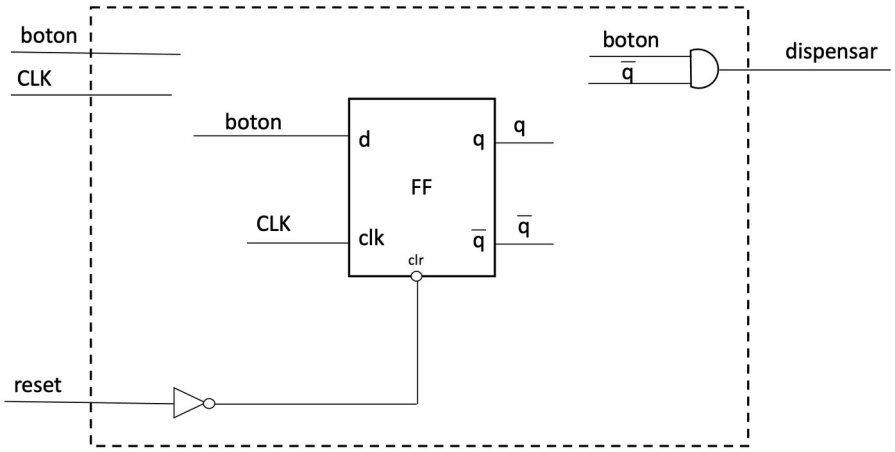
| q\boton | 0 | 1 |
|---|---|---|
| 0 | | 1 |
| 1 | | 1 |

d = b

| q\boton | 0 | 1 |
|---|---|---|
| 0 | | 1 |
| 1 | | |

dispensar = $b.\overline{q}$

7. Circuito



## *Pregunta 3*

En el modo de direccionamiento directo en la instrucción se encuentra la dirección del operando, mientras que en el modo de direccionamiento indirecto en la instrucción se encuentra la dirección de lugar donde se encuentra la dirección del operando.

En el caso de direccionamiento directo, se distinguen dos tipos:

- directo a registro: en este caso el operando está almacenado en un registro y la instrucción contiene el identificador de este registro. Por ende en este caso no se accede a memoria.

- directo a memoria: en este caso el operando está almacenado en memoria y la instrucción contiene la dirección donde se encuentra. Por ende en este caso se da un acceso a memoria, a la dirección mencionada, para traer el operando.

En el caso de direccionamiento indirecto, se distinguen dos tipos:

- indirecto por registro: en este caso el operando está almacenado en una posición de memoria cuya dirección se encuentra en un registro y la instrucción contiene el identificador del registro. Por ende, en este caso solo se realiza un acceso a memoria, con la dirección obtenida del registro.

- indirecto por memoria: en este caso el operando está almacenado en una posición de memoria cuya dirección esta en otra posición de memoria y la instrucción contiene la dirección de esta ultima. Por ende, en este caso se realizan dos accesos a memoria.

## *Pregunta 4*

Se denomina ciclo de instrucción a la secuencia de acciones que realiza la CPU (mas específicamente la Unidad de Control) para lograr ejecutar una instrucción del programa almacenado en memoria:

Un ciclo de instrucción típico tiene 5 pasos característicos:

- Fetch: este paso consiste en leer la próxima instrucción a ejecutarse desde la memoria.

- Decode: en este paso se analiza el código binario de la instrucción para determinar qué debe realizar (cual operación, con que operandos y donde guardar el resultado)

- Read: en este paso se accede a memoria para traer los operandos

- Execute: es la ejecución de la operación por parte de la ALU sobre los operandos.

- Write: en el ultimo paso se escribe el resultado en el destino indicado en la instrucción.

Para hacer la etapa de read la unidad de control genera las señales que controlan la lectura en memoria. Acciones:

1. Coloca la dirección del operando de forma que sea volcado en el bus de direcciones (p.e. cargando MAR).
2. Coloca en el bus de control la señal para indicar a la memoria una operación de lectura.
3. Se carga el operando de la memoria a través del bus de datos para que pueda ser usado (p.e. el contenido de memoria colocado en el bus de datos se almacena en MBR).

## *Solución Problema 1*

a)

```
short isPalindrome(char *str, short izq, short der) {
  short result;
     if (izq > der) result = true;
     else if (str[izq] != str[der]) result = false ;
     else result = isPalindrome(str, izq+1, der-1);
  return result
}
```

```
isPalindome proc
     pop AX         ; IP
     pop SI         ; DER
     pop DI         ; IZQ
     pop BX         ; str
     push AX

     cmp DI, SI
     jg isTrue
     mov CL, [BX+DI]
     cmp CL, [BX+SI]
     jne isFalse
     push BX
     inc DI
     push DI
     dec SI
     push SI
     call isPalindrome
     pop BX
     jmp fin
isTrue:
     mov BX,1
     jmp fin
isFalse:
     xor BX, BX
fin:
     pop AX
     push BX
     push AX
     ret
isPalindome endp
```

**b)**

Para un string de largo n se generan en el peor caso (cuando es palíndromo) techo(n/2) llamadas a la función.

La función retira los parámetros de entrada al inicio, no preserva contexto y acumula únicamente la dirección de retorno (IP) en las llamadas recursivas. El caso base acumula los parámetros de entrada y el IP. Entonces en el peor caso para un string de largo n se consume a lo sumo 2*techo(n/2)+8 bytes.

El string tiene un desplazamiento inicial de 0x9000 por lo que el string más largo que puede almacenarse en el segmento (DS) es 0x7000 caracteres. En este caso el consumo máximo será de 0x7008 bytes que puede almacenarse en el segmento de stack (SS).

## *Solución alternativa:*

**a)**

```
isPalindrome proc
    push bp
    mov bp, sp
    push ax
    push bx
    push cx
    push di
    push si

    mov bx, [bp+8] ; cargar puntero str
    mov si, [bp+6]  ; cargar izq
    mov di, [bp+4]  ; cargar der

; Comparar caracteres en índices izq y der
    cmp si, di
    jae ret_uno  ; si izq >= der, devuelve 1

    mov al, [bx+si]  ; cargar caracter en posición izq
    mov cl, [bx+di]  ; cargar caracter en posición der
    cmp al, cl
    jne ret_cero  ; si no son iguales, devuelve 0

; llamar a la función recursivamente
    inc si
    dec di
    push bx
    push si
    push di
    call isPalindrome
    pop ax
    jmp cleanup
```

```
ret_cero:
    mov ax, 0  ; la cadena no es palíndrome
    jmp cleanup

ret_uno:
    mov ax, 1  ; la cadena es palíndrome

cleanup:
    mov [bp+8], ax ; cargo el resultado en el stack
    mov bx, [bp+2]  ; guardo el ip
    mov [bp+6], bx  ; cargo el ip arriba del resultado
    pop si
    pop di
    pop cx
    pop bx
    pop ax
    pop bp
    add sp, 4   ; ajusto el sp para que apunte al ip
    ret
isPalindrome endp
```

**b)**

El consumo de stack en cada llamada es es de 4 palabras por el llamado (3 argumentos y la dirección de retorno) y 6 palabras por el contexto salvado. Esto vale tanto para la llamada inicial y el paso base, como para las llamadas recursivas.

Tendremos entonces que en el caso de un texto palíndromo de n caracteres, se realizarán techo(n/2) llamadas y el paso base, cada una consumiento 10 palabras = 20 bytes.

Dado que tenemos 64KB en cada segmento se deberá cumplir que:

(techo(n/2) + 1) * 20 bytes <= 65536 bytes

por lo que n deberá cumplir:

n < 6552

Por otro lado tenemos la restricción de la cantidad de bytes disponible en el DS para alojar el string, que ascienden a 0x10000 – 0x9000 = 0x7000 = 28762

En este caso vemos que es más restrictivo el consumo de stack y por tanto el máximo n es 6551.

## *Solución Problema 2*

```
#define OFF              0
#define ON_CALENTANDO    1
#define ON_ALCANZADA     2
#define TIMER_CALENTANDO 3
#define TIMER_ALCANZADA  4
#define ON               0x80

unsigned char deseada, delta, prendido, temporizador, minutos;
short int tics, indice, decilitrosxmin;
short int decilitrosxseg[60];

void interrupt timer() {

  unsigned char lectura;

  tics++
  lectura = in(CAUDAL);
  decilitrosxmin =  decilitrosxmin - decilitrosxseg[indice] + lectura;
  decilitrosxseg[indice] = lectura;
  indice = (indce + 1) % 60;
}

void interrupt configurar() {

  unsigned char lectura;

  lectura = in(TEMPERATURAS);
  deseada = lectura & 0x3F;
  delta = lectura >> 6;
  lectura = in(ENCENDIDO);
  prendido = lectura & 0x01;
  temporizador = (lectura >> 1) & 0x01;
  minutos = lectura >> 2;
}

void main() {

  unsigned char estado, temperatura, alarma;

  // instalo interrupciones
  alarma = OFF;
  indice = 0;
  for (indice = 0; indice < 59; indice++) decilitrosxseg[indice] = 0;
  decilitrosxseg[59] = CAUDAL_MINIMO * 10 + 1;
  decilitrosxmin = decilitrosxseg[59];
  estado = OFF;
  out(COMPRESOR, OFF);
  enable();
  while (ON) {
    temperatura = in(TEMP_AGUA);
    switch (estado) {
      case OFF:
        if (prendido)
          if (temperatura < deseada) {
            estado = ON_CALENTANDO;
```

```
      if (decilitrosxmin >= CAUDAL_MINIMO * 10) out(COMPRESOR, ON);
      else alarma = ON;
    }
    else {
      estado = ON_ALCANZADA;
    }
  break;
case ON_CALENTANDO:
    if (decilitrosxmin >= CAUDAL_MINIMO * 10) out(COMPRESOR, ON);
    else {
      out(COMPRESOR, OFF);
      alarma = ON;
    }
    if (temperatura >= deseada) {
      estado = ON_ALCANZADA;
      out(COMPRESOR, OFF);
    }
    if (temporizador) {
      tics = 0;
      estado = TIMER_CALENTANDO;
    }
    else if (!prendido) {
        estado = OFF;
        out(COMPRESOR, OFF);
    }
    break;
case ON_ALCANZADA:
    if (decilitrosxmin >= CAUDAL_MINIMO * 10) alarma = ON;
    if (temperatura < (deseada - delta)) {
      estado = ON_CALENTANDO;
      if (decilitrosxmin >= CAUDAL_MINIMO * 10) out(COMPRESOR, ON);
    }
    if (temporizador) {
      tics = 0;
      estado = TIMER_ALCANZADA;
    }
    else if (!prendido) {
        estado = OFF;
        out(COMPRESOR, OFF);
    }
    break;
case TIMER_CALENTANDO:
    if (decilitrosxmin >= CAUDAL_MINIMO * 10) out(COMPRESOR, ON);
    else {
      out(COMPRESOR, OFF);
      alarma = ON;
    }
    if (temperatura >= deseada) {
      estado = TIMER_ALCANZADA;
      out(COMPRESOR, OFF);
    }
    if (tics >= minutos * 60) {
      estado = OFF;
      out(COMPRESOR, OFF);
    }
    else if (!temporizador) {
      if (!prendido) {
        estado = OFF;
        out(COMPRESOR, OFF);
      }
      else estado = ON_CALENTANDO;
    }
```

```
          break;
      case TIMER_ALCANZADA:
          if (decilitrosxmin >= CAUDAL_MINIMO * 10) alarma = ON;
          if (temperatura < (deseada - delta)) {
            estado = TIMER_CALENTANDO;
            if (decilitrosxmin >= CAUDAL_MINIMO * 10) out(COMPRESOR, ON);
          }
          if (tics >= minutos * 60) {
            estado = OFF;
            out(COMPRESOR, OFF);
          }
          else if (!temporizador) {
            if (!prendido) {
              estado = OFF;
              out(COMPRESOR, OFF);
            }
            else estado = ON_CALENTANDO;
          }
          break;
      }
    out(DISPLAY, temperatura | alarma);
    }
  }
}
```