Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

El Lenguaje de Programación "C"

Una rápida revisión

Prof. Aurelio Jacinto Nolasco.
BUAP-FCE
Organización, Arquitectura de Computadoras P12

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Características del Lenguaje C

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

- Lenguaje de Propósito General.
- Programación Estructurada.
- Economía en expresiones.
- Abundancia en operadores y tipos de datos.
- Codificación en alto y bajo nivel.
- Producción de código objeto altamente optimizado.
- Facilidad de aprendizaje.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Estructura de un Programa en C

bloque de Directivas de Compilación.
bloque de Declaración de variables globales.
...
funcion1( )
{ *bloque de variables locales.*
  bloque de Implantación de algoritmo particular.
}
...
main( )
{ *bloque de variables locales.*
  bloque de implantación de algoritmo principal.
}

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Portabilidad del C.

- C es un lenguaje independiente del hardware y ampliamente disponible.
- Las aplicaciones escritas en C pueden ejecutarse con poca o ninguna modificación en diversos sistemas de cómputo.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Palabras Clave en ANSI-C

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipos de datos Simples o Fundamentales

Datos Enteros	char	signed char	unsigned char
	signed short int	signed int	signed long int
	unsigned short int	unsigned int	unsigned long int
Datos Reales	float	double	long double

En notación abreviada:

Datos Enteros	char	signed char	unsigned char
	short	int	long
	unsigned short	unsigned	unsigned long
Datos Reales	float	double	long double

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo char (*carácter*)

- **1 Byte.**
- **Contiene un único carácter.**
- **Valor entero de -128 a 127 correspondiente a un carácter del código ASCII** (*solamente los valores de 0 a 127 son equivalentes a un carácter*).
- **unsigned char *puede almacenar de 0 a 255.***

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo short
(*entero en formato corto*)

- 2 Bytes.
- Entero en el rango -32 768 a 32 767 [-2^15, 2 ^ 15 -1].
- unsigned short almacena en el rango 0 a 65 535 [0, 2 ^ 16 -1].

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo int
(*números de tipo entero*)

- Número sin punto decimal.
- El rango de valores depende de la máquina y del compilador.
- Normalmente ocupan 2 Bytes.
- Cuando se le asigna en la ejecución un valor fuera del rango permitido (overflow), se produce un error en el resultado (no avisa)

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo long
(*entero con formato largo*)

- 4 Bytes (puede variar)
- Rango -2 147 483 648 a 2 147 483 647
  [-2e31, 2e31 -1]
- Para **unsigned long** el rango es [0, 2e32 -1]
- En general tamaño(**int**) < tamaño(**long**)

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo float

(*Números reales de simple precisión*)

- Números con una parte entera y una fraccionaria también llamados números de punto ó coma flotante.
- Se representan por la mantisa y un exponente (potencia de 10 por la que se multiplica la mantisa)
- pi = $0.3141592654 \times 10^{1}$
- mantisa y exponente pueden ser positivos o negativos

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo float

- 4 Bytes.
- 24 bits para la mantisa (1 para el signo y 23 para el valor)
- 8 bits para el exponente (1 para el signo y 7 para el valor)
- Precisión: número de cifras con las que se representa la mantisa 2^{23} =8 388 608 entre 6 y 7 cifras de precisión
- Rango:

  [1.175494e-38, 3.402823e38] para números positivos

  [-3.402823e38, -1.175494e-38] para números negativos

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo double
(*reales con doble precisión*)

- 8 Bytes
- 53 bits para la mantisa y 11 para el exponente
- hasta 16 bits de precisión.
- Rango:

  [-1.79769e308, -2.22507e-308] para negativos

  [2.22507e-308, 1.79769e308] para positivos

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo long double

(*reales doble precisión formato largo*)

- 10 Bytes
- Entre 18 y 19 bits de precisión
- 64 bits para la mantisa y 16 para el exponente
- Rango

  [-1.189731e4932, -3.362103e-4932]

  [3.362103e-4932, 1.189731e4932]

  – Depende del compilador y de la máquina.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipo void

- Para declarar funciones que no retornan un valor
- para declarar un apuntador a un tipo no especificado
- para especificar que una función no acepta argumentos.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Resumen Tipos de Datos más utilizados.

Nombre	Bytes	Min	Max
(signed) char	1	-128	127
unsigned char	1	0	255
(signed) (short) int	2	-32768	32767
unsigned (short)(int)	2	0	65535
(signed) long (int)	4	-2147483648	2147483647
unsigned long (int)	4	0	4294967295
float	2	3.4E-38	3.4E+38
double	4	1.7E-308	1.7E+308
long double	5	3.4E-4932	1.1E+4932

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Estructuras de Control para Flujo de los datos en lenguaje “C”.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

- *Instrucciones **Secuenciales:*** Las Instrucciones de un programa en C, se ejecutan una detrás de la otra.
- *Instrucciones **Selectivas:*** permiten elegir entre dos o más opciones, según ciertas condiciones.
- *Instrucciones **Repetitivas:*** permiten ejecutar repetídamente un conjunto de instrucciones, tantas veces como se desee, cambiando o actualizando ciertos valores.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Notación

- **Proceso**: representa una instrucción simple o compuesta.
  - Cada instrucción simple debe estar separada de la anterior por un **;**.
  - La instrucción compuesta inicia con una llave ***{*** y termina con otra ***}*** **(estructura secuencial).**

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Estructuras de Control en lenguaje C

- Selectivas o Toma de Decisión.
  - Instrucción *if* …;
  - Instrucción *if* ... *else* …;
  - Instrucción *switch* ... *case* …:…*default*: …;
  - *Operador condicional ?...:…;*
- Repetitivas o Lazos
  - Instrucción *while* ...;
  - Instrucción *do* ... *while;*
  - Instrucción *for*(…;…;…)…;
- Instrucciones *break*, *continue*

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Notación

- Proposición (PROP) (**Expresión booleana**): debe ser una ***expresión numérica***, ***relacional*** o ***lógica***.

  Los posibles resultados que se obtienen al evaluar la proposición, son:

  – Verdadero (***distinto de cero***)
  – Falso (***igual a cero***)

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Instrucción *if…;*

…

if (PROP)

proceso1;

…

Se evalúa la **expresión booleana** (PROP).

- Si el resultado es *verdadero*, se ejecuta el proceso1 y se continúa.
- Si el resultado es *falso*, se salta el proceso1 y se continúa.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Instrucción *if* …; *else*…;

```
...
if (PROP)
   proceso1;
else
   proceso2;
...
```

Se evalúa la **proposición booleana** (PROP).

- Si el resultado es verdadero, se ejecuta el proceso1 y **no** se realiza el proceso2 y continúa.
- Si el resultado es falso, se salta el proceso1 y se ejecuta el proceso2 y continúa.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operador ternario (*Operador Condicional*)

...

(PROP) *?* proceso1*:* proceso2;

...

Se evalúa la **expresión booleana.**

- Si el resultado es verdadero, se ejecuta el proceso1
- si el resultado es falso, se ejecuta el proceso2.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Instrucción *while*…;

```
...
while(PROP)
   proceso1;
...
```

- Mientras la expresión (PROP) es verdadera se ejecuta el proceso1; esto es, se repite el proceso1 mientras (PROP) es verdadera.
- Si (PROP) es falsa, **NO** se ejecuta proceso1, saliendo entonces del proceso while, continuando con la siguiente instrucción.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Instrucción *do ... While ;*

```
...
do
   proceso1;
while(PROP);
...
```

Se ejecuta al menos una vez el proceso1; el proceso1 se vuelve a ejecutar mientras (PROP) es verdadera.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Instrucción *switch...case…*

```
...
switch(ExpNum)
{case const1:proceso1;
  ...
  case constN:procesoN;
default: procesoD;
}
...
```

Esta instrucción es ***SELECTIVA***.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Instrucción *for*(;;)…;

```
...
for(proceso1;PROP;proceso3)
   proceso2;
...
```

Inicialmente se ejecuta el proceso1; se evalúa la PROP; si es verdadera, se ejecuta el proceso2, inmediatamente el proceso3. Si la PROP es falsa, entonces se continúa.

Esta instrucción es REPETITIVA.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operadores y Expresiones

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operador de Asignación

Asignación simple =

Operadores Aritméticos

Suma	+	(enteros, reales)
Resta	-	(enteros, reales)
Multiplicación	*	(enteros, reales)
División	/	(enteros, reales)
Residuo o Módulo	%	(enteros)

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operadores de Incremento

Incremento ++ (prefijo, Postfijo)
Decremento -- (prefijo, Postfijo)

Operadores **unarios** que aumentan o disminuyen en una unidad.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operadores de Asignación Compuestos

Suma más asignación	+=
Resta más asignación	-=
Multiplicación más asignación	*=
División más asignación	/=
Módulo o resto más asignación	%=
Desp. a la izq. más asignación	<<=
Desp. a la der. más asignación	>>=
And sobre bits más asignación	&=
Or sobre bits más asignación	|=
Xor sobre bits más asignación	^=

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operadores Relacionales.

Igual que	==
Menor que	<
Mayor que	>
Menor o igual que	<=
Mayor o igual que	>=
Distinto de	!=

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operadores Lógicos
(argumentos lógicos)

And	&&
Or	||
Not	!

Operadores unarios
(argumentos numéricos)

Cambio de signo	-	(entero, real)
Complemento a 1	~	(entero, ASCII)

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Operadores lógicos para manejo de bits. (argumentos Numéricos resultado Numérico)

And (Y) de bits	&
Or (O) de bits	|
Xor (O exclusiva) de bits	^
Not (complemento a 1)	~
Desplazamiento a la izquierda	<<
Desplazamiento a la derecha	>>

*Todos con Operandos enteros.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Expresiones Aritméticas

- Análogas a fórmulas matemáticas
- Utilizan operadores aritméticos e incrementales

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Expresiones Lógicas o Booleanas

- Su resultado es Verdadero (diferente de cero) o Falso (igual a cero)
- Sus argumentos son valores lógicos y los operadores *lógicos* y/o *relacionales*.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Expresiones Generales

- C es flexible para combinar expresiones y operadores de distintos tipos en una expresión
- Cualquier expresión lógica puede aparecer como sub-expresión en una expresión aritmética.
- Cualquier expresión aritmética puede aparecer como sub-expresión en una expresión lógica.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Precedencia de Operadores

Precedencia (categoría)	Operador
1 La más Alta	( ), [ ]
2 Unarias	!, ~, +, -, ++, --, &, *, sizeof( )
3 Multiplicativos	*, /, %
4 Aditivos	+, -
5 Corrimientos	<<, >>
6 Relacionales	<, <=, >, >=
7 Igualdad	==, !=
8	&
9	^
10	|
11	&&
12	||
13	?:
14 Asignación	=, *=, /=, %=, +=, -=, &=, ^=, |=, <<=, >>=

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

(I/O)

Biblioteca Standard

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Funciones predefinidas en C

- C dispone en sus bibliotecas, con más de 400 funciones (199x).
  - Entrada y Salida {Lectura/Escritura} (I/O).
  - Manejo de cadenas de caracteres (String).
  - Matemáticas.
  - Clasificación de caracteres.
  - Funciones relativas a tiempos.
  - Funciones relativas a manejo de Puertos.
  - Etc.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Bibliotecas

- Las funciones ya compiladas, pueden ser guardadas en archivos de Bibliotecas.
- Las Bibliotecas son conjuntos de funciones compiladas, normalmente con una finalidad análoga relacionada, que se guardan en un archivo bajo un determinado nombre, que están listas para ser utilizadas por cualquier usuario.
- Para poder ser utilizadas, es necesario hacer uso de la ***Directiva de Compilación***:

#include <nombre.h > ó **#include “nombre.h”**

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Funciones de Entrada/Salida

- El lenguaje C ***NO*** dispone de instrucciones de entrada/salida.
- Para ello se utilizan funciones contenidas en la biblioteca estándar.
- Son un conjunto de funciones, incluidas con el compilador, que permiten a un programa recibir y enviar datos al exterior.
- Para utilizarlas, es necesario *incluir* el archivo de biblioteca ***stdio.h***, a través de la *directiva de compilación*:

```
#include <stdio.h>
```

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Función de **Escritura** *printf( )*

- Imprime en la ***unidad de salida*** (por default, el monitor), el texto, constantes y variables que se indiquen.

int printf(“cadena_de_control”, tipo arg1, tipo arg2, ...)

  - cadena_de_control especifica el formato en que aparecerán los argumentos.
  - arg1, arg2, ... Representan los valores a ser escritos.

- Una cadena de control se compone de:

%[flags] [ancho] [.precisión][{h/l/L}] tipo.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Función de I/O *scanf( )*

- Es análoga a *printf()*
- Lee datos de la entrada estándar (el teclado), los interpreta de acuerdo al formato indicado y los almacena en los argumentos especificados.
- Se encuentra en la biblioteca ***stdio.h***
- Su forma general es:

int scanf(“%x1 %x2 ...”, &arg1, &arg2 ,...);

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Función *scanf( )*

- arg1, arg2 son *apuntadores* a las variables que se quieren leer.
- %x1, %x2 son las *especificaciones de formato* que interpretan cada dato de entrada y separadores (espacios en blanco).
- La cadena de control o especificación de formato está compuesta:

%[*] [ancho] [{h/l/L}] tipo.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

(I/O)

Archivos

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Tipos de Archivos

- Los archivos pueden ser ***de lectura, de escritura o de lectura/escritura.***
- Por el tipo de datos
  - Texto
  - Binarios
- Por el tipo de acceso
  - Secuenciales
  - Aleatorios

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

- Archivos Texto
  - Lo que se lea o escriba, es interpretado como código ASCII (a través de los tipos de datos básicos, números, caracteres, etc…)
- Archivos Binarios
  - Lo que se lea o escriba, es interpretado como código Binario.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Archivos Secuenciales

- Todo lo que se escribe (para archivos de escritura) se va “pegando” al final del archivo (sean del tipo Texto o Binario).
- Todo lo que se lee (para archivos de lectura) se va “sacando” del inicio del archivo hacia el final.

Este tipo de archivos son muy útiles. ej. Para la transferencia de información entre sistemas (cómputo, programas), ej. P/gráficos.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Archivos Aleatorios

- Para los de lectura y/o escritura ya sean texto o binario, es definido un apuntador que podemos manipular para asignar ó recuperar una posición dentro del archivo.

Este tipo de archivos son muy útiles ej. En el desarrollo de bases de datos.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Funciones para Archivos en C

- En el lenguaje C, en la biblioteca de funciones estándares (stdio.h) existen funciones para:
  - Crear (create)
  - Abrir (open)
  - Cerrar (close)
  - Borrar (delete)
  - Buscar (seek) (dentro del archivo)
  - Fin de Archivo? (EOF)
  - Etc.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Funciones Lectura/Escritura para Archivos en C

- Funciones para leer o escribir información en un archivo tales como:
  - Imprimir con formato (fprintf()).
  - Leer con formatos (fscanf()).

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

(I/O)

Puertos

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Puertos

- Punto de comunicación con el exterior.
  - Salida *(O)*
  - Entrada *(I)*
  - Entrada - Salida *(I/O)*

* ***Seriales.***

- 1 bit a la vez.
- Síncrono ó Asíncrono.

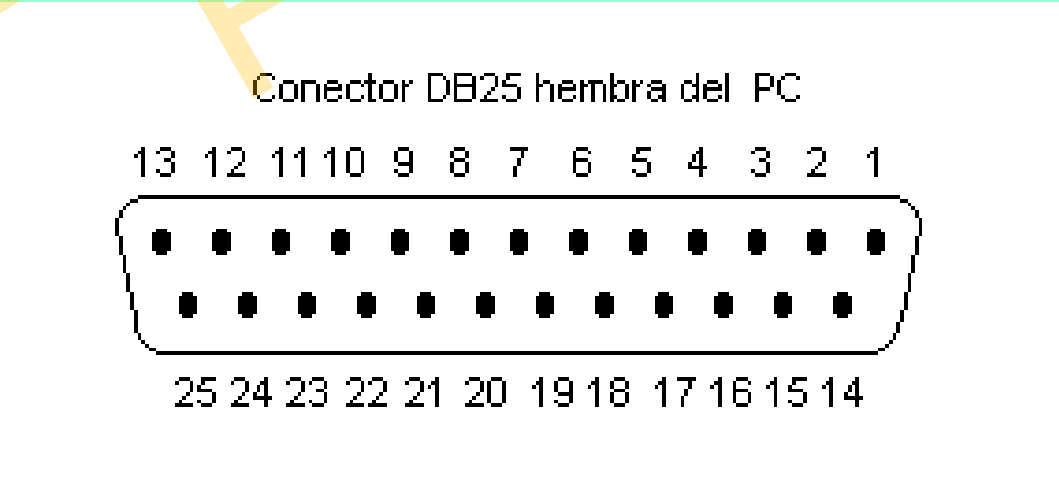
* ***Paralelos.***

- 8 bits a la vez (normalmente 2^n).
- Síncrono.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

El puerto Paralelo
(para la impresora)

- ***El Puerto Paralelo para la Impresora.***
  - Tiene una dirección **BASE.** (378h, ó 3BCh)
  - Se encuentra en un conector de la PC tipo DB25 hembra.

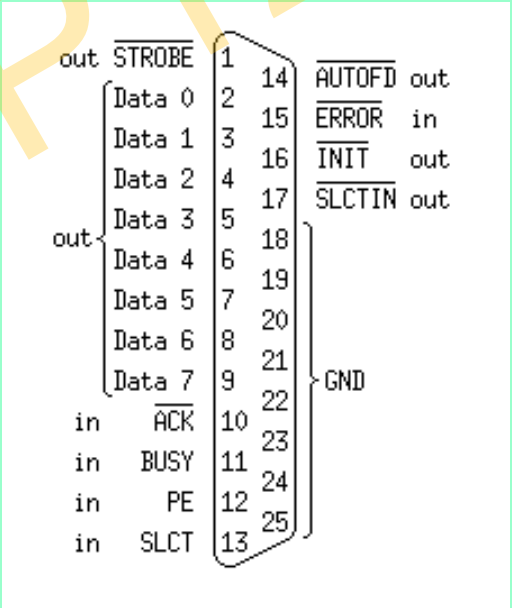


Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

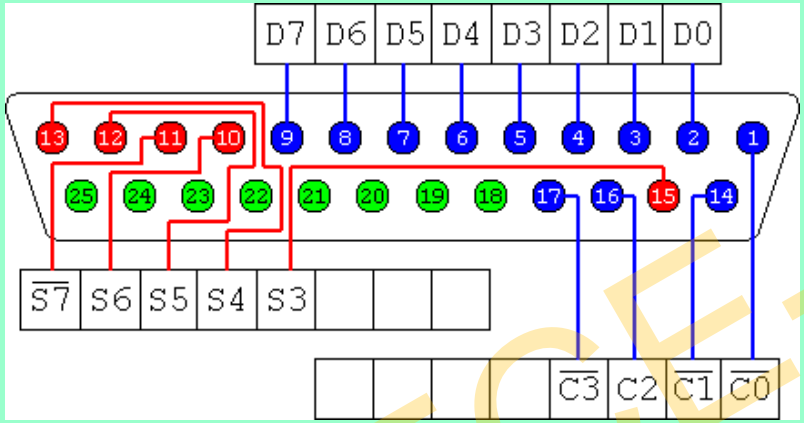
El puerto Paralelo

- El puerto se divide en tres Registros
  - DATOS. *(Escritura/Lectura)* **BASE**
  - CONTROL. *(Escritura/lectura)* **BASE+1**
  - STATUS. *(Lectura)* **BASE+2**

  - *Escritura* ***Computadora ------> Exterior***
  - *Lectura* ***Computadora <------ Exterior.***



Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx



El Puerto Paralelo

- **El *registro* de DATOS.**
  - Registro bidireccional.(*I/O*).
  - 8 bits
  - LATCH.

D7	D6	D5	D4	D3	D2	D1	D0

  - Todos los bits activos en ALTO.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

El Puerto Paralelo para la Impresora.

- **El *registro* de CONTROL.**
  - Registro bidireccional.(*I/O*).
  - 4 bits.
  - LATCH.
  - Salida a Colector abierto.

X	X	X	X	C3	C2	C1	C0

  - Todos los bits activos en BAJO excepto C2.
  - Los bits X no se utilizan.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

El Puerto Paralelo para la Impresora.

- **El *registro* de STATUS.**
  - Registro entrada.(*I*).
  - 5 bits
  - LATCH.

S7	S6	S5	S4	S3	X	X	X

  - Todos los bits activos en ALTO excepto S7.
  - Los bits X no se utilizan.

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Acceso a los Registros a través de C.

- Macro Funciones en lenguaje C.
  - int inportb(int dirección)
  - int inport(int dirección)
  - void outportb(int dirección,int dato)
  - void outport(int dirección,int dato)

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Mapa de Direcciones de Dispositivos

0x00000000-0x00000CF7	Controladora de acceso directo a memoria	OK
0x00000010-0x0000001F	Recursos de la placa base	OK
0x00000020-0x00000021	Controladora de interrupciones programable	OK
0x0000002E-0x0000002F	Recursos de la placa base	OK
0x00000040-0x00000043	Cronómetro del sistema	OK
0x0000004E-0x0000004F	Recursos de la placa base	OK
0x00000060-0x00000060	Quick Launch Buttons	OK
0x00000061-0x00000061	Altavoz del sistema	OK
0x00000062-0x00000062	Controladora incrustada compatible con Microsoft ACPI	OK
0x00000064-0x00000064	Quick Launch Buttons	OK
0x00000066-0x00000066	Controladora incrustada compatible con Microsoft ACPI	OK
0x00000070-0x00000071	Sistema CMOS/reloj en tiempo real	OK
0x00000072-0x00000073	Sistema CMOS/reloj en tiempo real	OK
0x00000080-0x0000008F	Controladora de acceso directo a memoria	OK
0x00000092-0x00000092	Recursos de la placa base	OK
0x000000A0-0x000000A1	Controladora de interrupciones programable	OK
0x000000B0-0x000000B1	Recursos de la placa base	OK
0x000000C0-0x000000DF	Controladora de acceso directo a memoria	OK
0x000000F0-0x000000FF	Procesador de datos numéricos	OK
0x00000378-0x0000037F	Puerto de impresora ECP (LPT1)	OK
0x000003B0-0x000003BB	Puente PCI estándar de PCI a PCI	OK
0x000003B0-0x000003BB	Integrated ATI MOBILITY RADEON X300	OK
0x000003C0-0x000003DF	Integrated ATI MOBILITY RADEON X300	OK
0x000003F6-0x000003F6	Canal IDE principal	OK
0x00000778-0x0000077A	Puerto de impresora ECP (LPT1)	OK

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Vista del 2do Kb de la RAM
Base de Datos BIOS

Prof. Aurelio Jacinto Nolasco
ajacinto@ece.buap.mx

Gracias