

19/3/15

log. Taller

Técnicos Digitales II. (teoría)

Bibliografía en la web de digitales II.

Hoy en la electrónica casi todo es basado en sistemas con micro, con diferentes características. Ej: teléfonos móviles, hornos microondas, automóviles, etc.

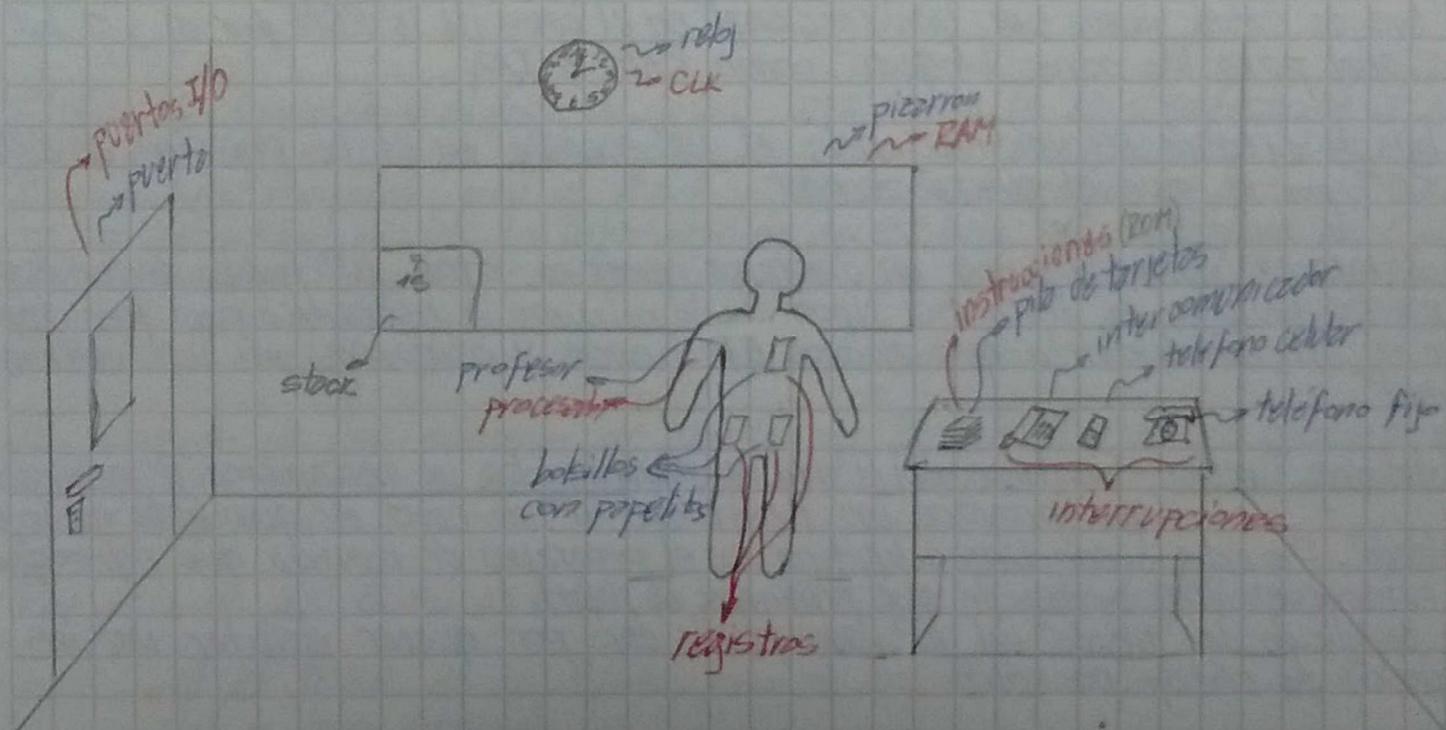
En general, todos los micros tienen muchísimas similitudes. Y en particular, se ven sus diferencias entre ellos.

Estas similitudes o diferencias, son de "forma" o de "fondo". Las cuestiones de fondo en los micros, tienen similitudes; en formas están las diferencias.

Memoria Cache: memoria de acceso rápido, que guarda temporalmente la información. Además es volátil (del tipo RAM).

Hoy memorias más lentas y otras más rápidas. Los más rápidos son más caros. El hecho de que sea más caro o barato viene dado por la construcción interna de la memoria, por ej. que transistores uso, el área que ocupa, cuántos se usan, etc.

Analogía del μP: definimos una situación conocida que se desarrolla en forma similar a como trabaja el μP.



El μP lee y ejecuta las instrucciones no en cualquier momento, sino cuando se lo indica el clock, se dice que esto es sincronizado.

Los teléfonos representan las interrupciones que tienen distintos jerarquías. Adoptamos por convención si hay una interrupción durante una ejecución de una

instrucción, primero se realiza la instrucción, segundo se anota en el stack la dirección de la siguiente "tarjeta", tercero atiendo la interrupción.

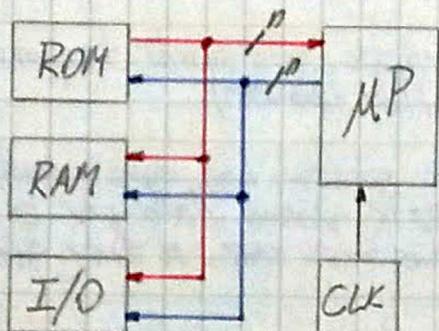
Las interrupciones serán nuevas instrucciones para ejecutar. Al finalizar éstas instrucciones, voy a leer el stack para retomar lo que estaba haciendo antes de la interrupción.

Las interrupciones se clasifican en internas y externas.

Un sistema basado en micro se comporta más o menos como éste anotado.

El conjunto de tarjetas representan el programa, y el número de ellas es la dirección de las instrucciones.

Diagrama General de un sistema basado en micro.



- Data Bus (puede ser unidireccional en el caso ROM o bidireccional)
 - Address Bus (unidireccional)
- Capacidad de buses (Número de bits)

26/3/15

El bus de datos puede ser unidireccional en el caso de la ROM por ej., o bidireccional para los otros bloques, como RAM e I/O.

El bus de dirección es unidireccional; el MP "escribe" en él la posición de RAM o ROM por ej. que él quiere acceder.

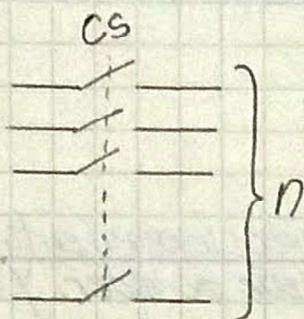
El Diagrama General es eso un diagrama genérico; otros MP pueden tener más o menos bloques de distintos tipos. Es lo mínimo y necesario para funcionar. Por ej. un complemento que algunos MP tienen y otros no, es un "watchdog timer".

Watchdog Timer: Contador del Perro del Guardia. Es un mecanismo de seguridad que provoca un reset del MP en caso de que éste se haya "tildado". Se debe crear una subrutina en el programa de manera que refresque o reinicie el WDT antes de que provoque el reset. La condición de "tildado" del MP, se puede dar por querer ejecutar una instrucción de código no válido.

El set o conjunto de instrucciones del MP, tiene instrucciones válidas y otras no válidas o "ocultas", donde su ejecución puede tener resultados impredecibles.

El bus de datos y el bus de direcciones comunican a todos los bloques, para que eso sea posible y sobre todo en el bus de datos que es bidireccional.

deberán tener la característica de 3º Estado, o sea que los dispositivos podrán poner sus puertos en alta impedancia, lograr una "desconexión" cuando no se los selecciona por medio de un "chip select".



Se habilitará un dispositivo a la vez de los que están conectados al bus.

El MP selecciona qué dispositivo quiere leer por ej., y por el bus se envía la información al MP.

El mapeo de memoria consiste en cómo decodificar de acuerdo a la cantidad de memoria que se tiene en el sistema para que se habilite o no a una determinada dirección de memoria.

Por ejemplo: desde la posición 300H quiero cobzar una memoria de 1K ($1K = 1024$ direcciones)

$d = 300H \rightarrow 0011\ 0000\ 0000$ }
| | |
1024 posiciones - 1K
 $d = 0FFH \rightarrow 0110\ 1111\ 1111$ }
| | |
= 400H
 $d = 700H \rightarrow 0111\ 0000\ 0000$

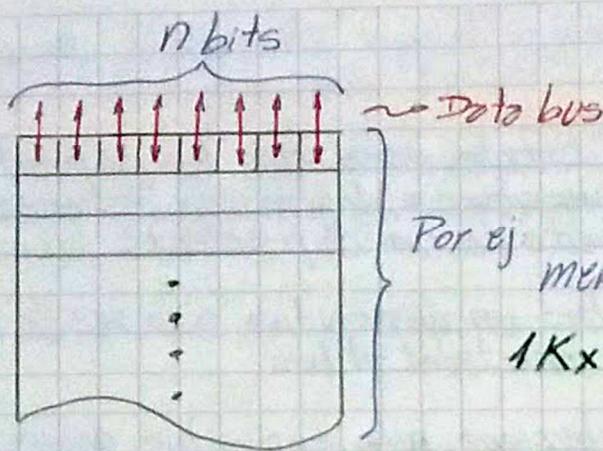
El bus de dirección y el bus de datos, me permite la clasificación de los MP. Por ej. el bus de datos se clasifican en 8bit, 16, 32, 64 bits. Con un bus pequeño, por ej. 8bit y necesito manejar un dato de 32bits, voy a necesitar enviar la información de 32bits en 4 tramos de 8bits. Esto insume tiempo, comparado si mi bus fuese de 32bits, se hace un solo envío del dato.
Entre más "ancho" el bus de datos, más es la velocidad.

• El producto de dos números de n bits es 2^n bits. El máximo número en 2 bits es 3 (11b), si se hace $3 \times 3 = 9$ la representación en binario es 1001, cuatro bits.

Por ej. • el MP 8086 de Intel es de 16 bits de bus de datos externo.

• el MP 8088 es de 8 bits, pero de arquitectura interna de 16 bits.
(bus de datos interno)

Esquemáticamente se puede pensar a la memoria como:



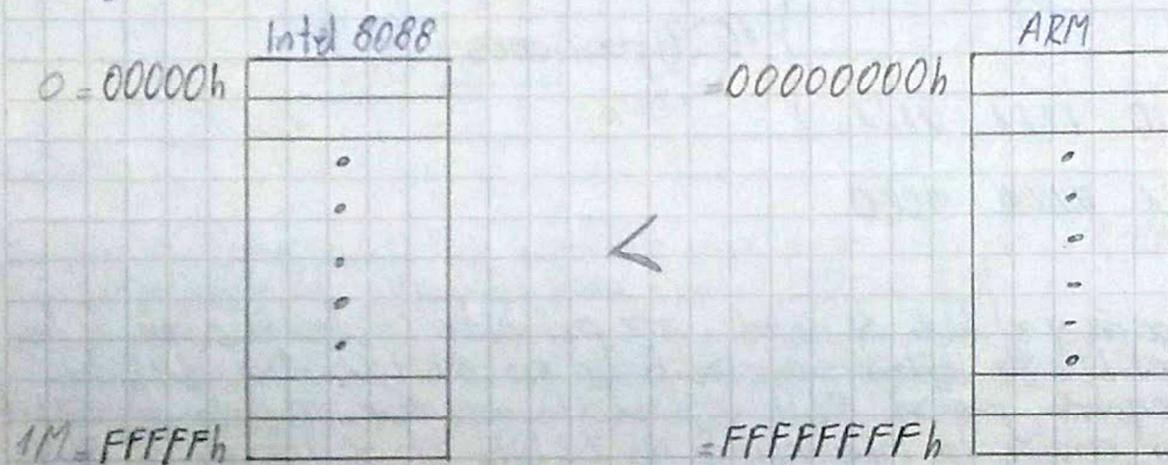
Faltó definir mapeo espejo
 - ventajas/desventajas de la diferencia de tamaño del bus interno y externo
 - Modo Real y Modo Protegido

El bus de dirección le da la capacidad a los, la cantidad de direcciones de memoria que fija por la cantidad de líneas del bus address.

Por ej. el μP de Intel tiene 20 líneas de bus address, puede direccionar $2^{20} = 1M$ posiciones de memoria. El ARM tiene 32 líneas, $2^{32} = 4G$ posiciones de memoria.

El mapa de memoria es la representación de todas las posiciones de memoria que el μP puede acceder.

Por ej.



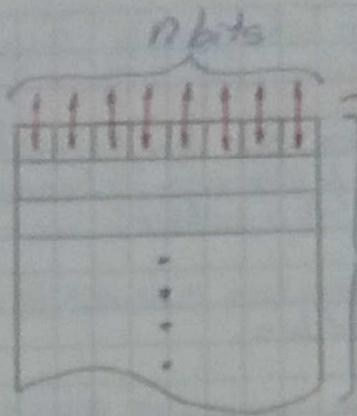
Todos los μP de Intel tienen 2 modos, uno el modo real y el otro el modo protegido. Los μP arrancan en modo real y luego el SO lo cambia a modo protegido.

Los μP trabajando en modo real se comportan como el 8088; si un SO no conmuta el μP a modo protegido, uno tendría un i7 como un 8088 por ejemplo.

Los μP Intel acceden a la memoria en forma segmentada, osea por porciones.

Los ARM no tienen segmentación. - acceden en forma aleatoria

- Es difícil a pensar como un auto de carreras, con sus bancos vacíos, y te das todos posiciones libres para entrarle a "acceder". También one puede entrar sin miedo de "bajar" (por ej.) y tener acceso a ellos porque estos son bancos, pero no si resto de los bancos, eso sería de forma segmentada.



~ Data bus

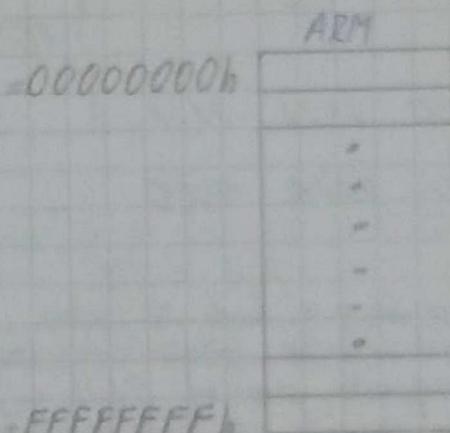
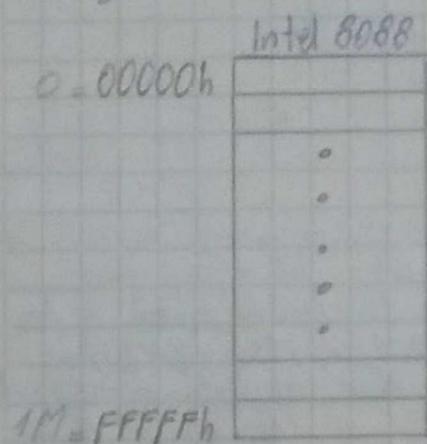
Por ej. memoria de 4096 posiciones de 8 bits
1Kx8

El bus de dirección le da la capacidad a los μP de "direcciónamiento", es la cantidad de direcciones de memoria que el μP puede acceder. Y esto fijado por la cantidad de líneas del bus dirección.

Por ej. el μP de Intel tiene 20 líneas de bus dirección, puede direccionar $2^{20} = 1M$ posiciones de memoria. El ARM tiene 32 líneas, $2^{32} = 4G$ posiciones de memoria.

El mapa de memoria es la representación de todas las posiciones de memoria que el μP puede acceder.

Por ej.



Todos los μP de Intel tienen 2 modos, uno el modo real y el otro el modo protegido. Los μP arrancan en modo real y luego el S.O. lo cambia a modo protegido.

Los μP trabajando en modo real se comportan como el 8088, si un S.O. no commuta el μP a modo protegido, uno tendría un i7 como un 8088 por ejemplo.

Los μP Intel acceden a la memoria en forma fragmentada, o sea por porciones.

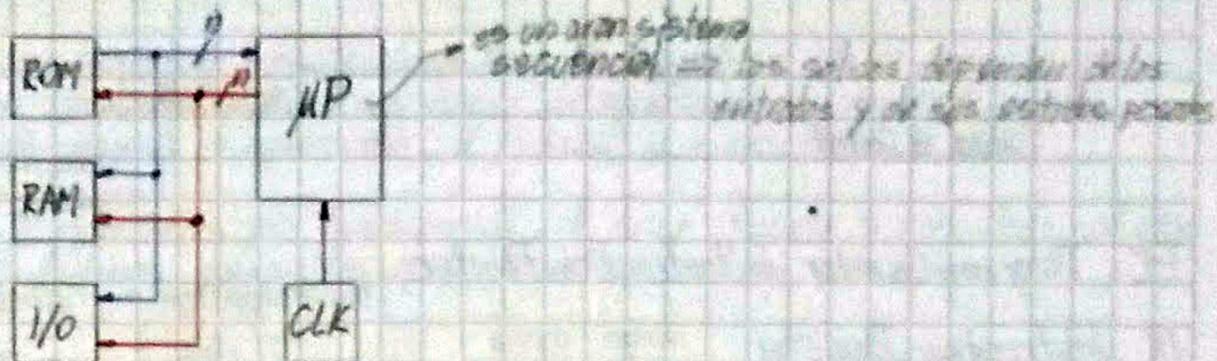
Los ARM no tienen segmentación - acceden en forma continua.

- Es más fácil a pensar, como una auto de circuito con sus casillas vueltas, ya tiene todos los lugares llenos para saltar de una a la otra. Tienen un gran espacio en modo de Rendimiento (por ej.) y tienen acceso a otras porciones de memoria. Pero no al resto de la memoria, esa sería de fabrica segmentada.

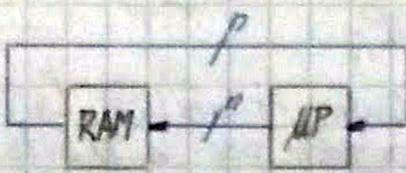
9/4/15

- El hacer diagramas con bloques son muy importantes para captar ideas y expresar técnicamente esas ideas.

Hasta aquí, vimos un diagrama en bloques de un sistema basado en micro.



- Puedo prescindir del 3º estado para conectar por el databus el MP y memoria?
- Si. No es lo más eficiente, pero es posible.



Para acceder a más bloques de memoria, deberás repetir este esquema y de modo similar implementar un combinacional para elegir la RAM x y su posición de memoria (mapeo).

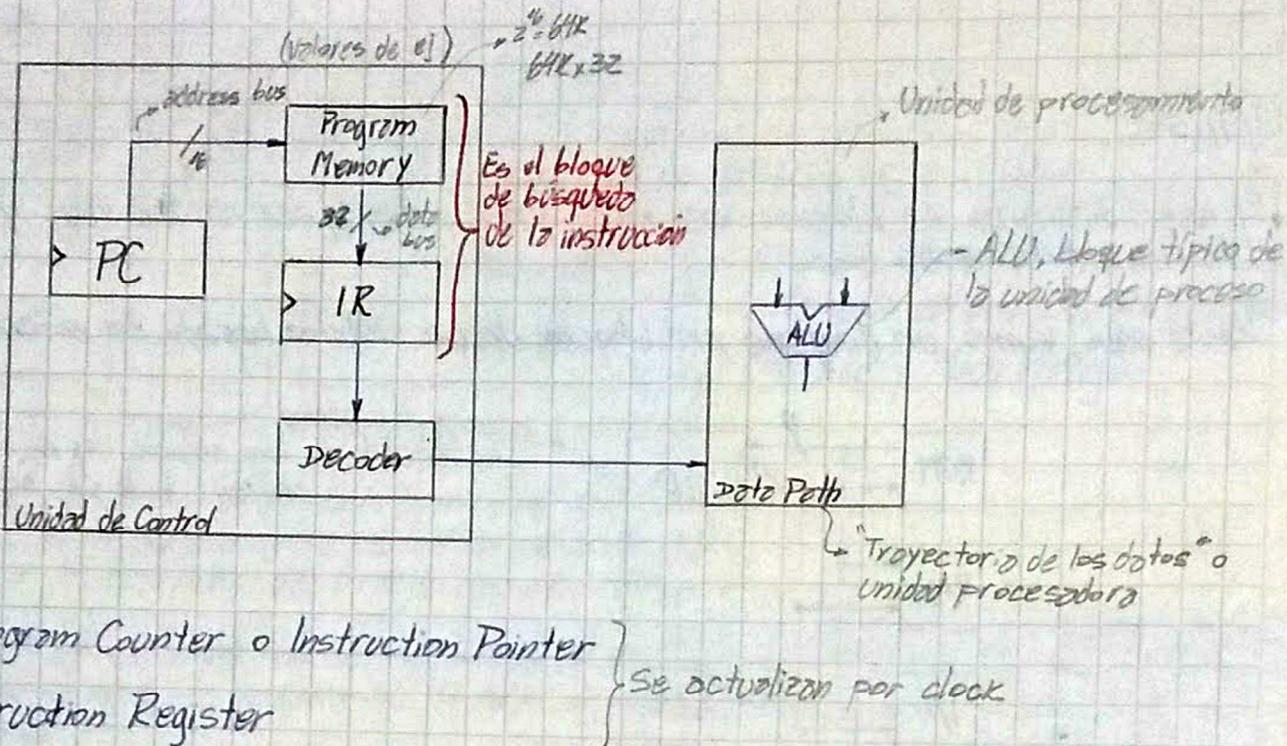
- La capacidad de direccionamiento de un MP lo indica la cantidad de líneas (bits) del bus de dirección.

Ejemplo: Intel 8086 \rightarrow 20 líneas $\rightarrow 2^{20} = 1M$

ARM \rightarrow 32 líneas $\rightarrow 2^{32} = 4G$

- Cuáles son las operaciones básicas que realiza un MP?

1. Busqueda de la instrucción.
2. Decodifica la instrucción. (interpreta la instrucción)
3. Ejecuta la instrucción.



PC: Program Counter o Instruction Pointer

IR: Instruction Register

Unidad de procesamiento

- ALU, Bloque típico de la unidad de proceso

"Troyectorio de los datos" o
Unidad procesadora

- La cantidad de bits del PC depende de la cantidad de líneas del bus de dirección. (son cuales)
- El código de la instrucción depende de los **MP**.

En el Data Path se encuentra la **ALU** (Unidad Lógica Aritmética). La **ALU** tiene dos operandos y un resultado; los 3 casos pueden ser de n bits. También existen "banderas" o "Registro de Estado": "Program Status Word". Muestro como fueron las operaciones.

S	Z	V	C

C: carry

Z: zero si el resultado es cero

V: overflow

S: sign (signo) en base al bit más significativo del resultado

- Los números negativos se representan por el complemento a 2.
Ej. -5 expresado en 8bits.

Para saber el -5 o representar, y lo escribo como positivo, 5

$$5 = \begin{array}{r} 0\ 0\ 0\ 0\ 0\ 1\ 0\ 1 \\ 1\ 1\ 1\ 1\ 0\ 1\ 0 \\ \hline \end{array} \text{negado}$$

- Sumar 2 números positivos o sumar 2 números negativos se puede ir fuera de rango el resultado
- Sumar 2 números (+, -) o (-, +) no se va fuera de rango, pero tengo acarreo.

- Ejemplo: sumar un n° positivo con un negativo

$$\begin{array}{r} + 01 \dots \\ 11 \dots \\ \hline 1100 \\ C \end{array}$$

tengo un acarreo, y el número de resultado no está fuera de rango.

- Todo registro es un conjunto de Flip-Flops. Por ej. el **Program Status Word**, sus

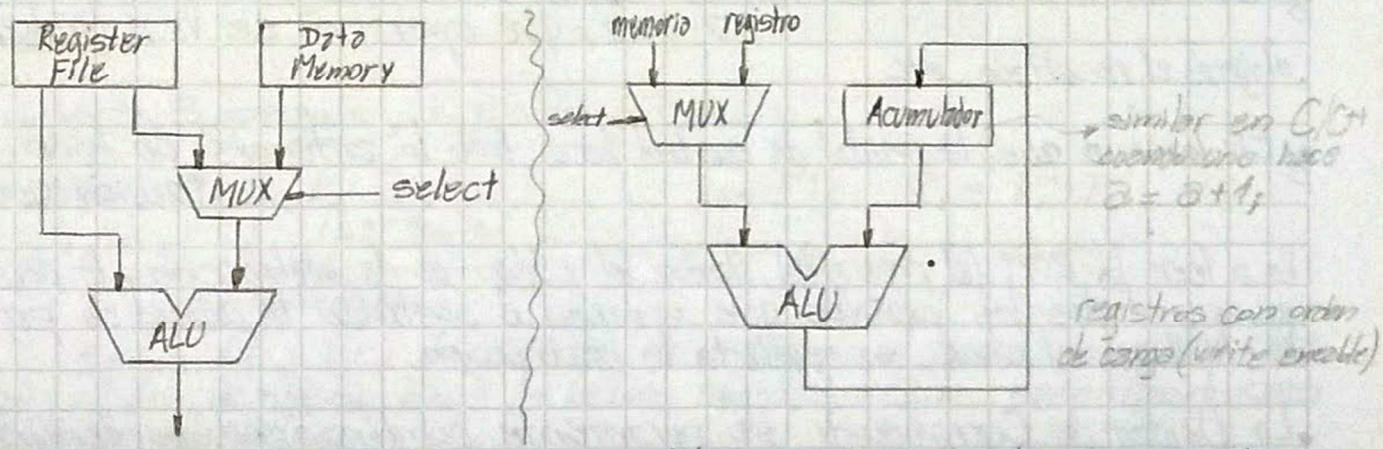
"bomberos" son los salidas de un FF; este registro está gobernado por el clock, o sea que se actualiza por el clock.

- Quienes le suministran los operandos a la ALU? Son los registros

No todas las ALUs operan con registros solamente, también pueden operar con registros y memoria. Varían de acuerdo a su construcción.

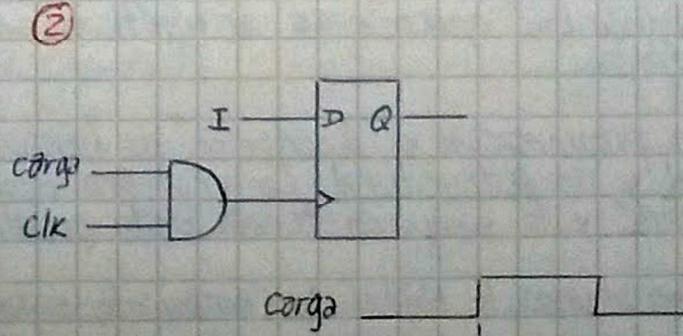
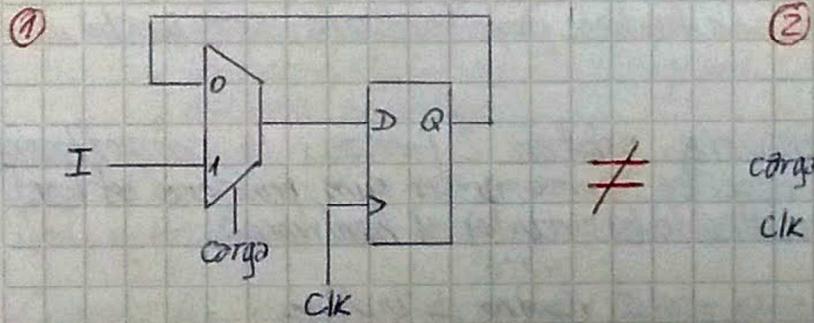
- ARM opera con registros; si quisiera con memoria, el dato de memoria debió moverlo a un registro, para luego hacer la operación registro con registro. Cada instrucción maneja 3 operandos, 2 fuentes y 1 resultado.

- Intel maneja 2 operandos por instrucción, 1 será fuente y resultado, el otro solo fuente. Además permite que algún operando sea memoria.



- Estos diseños se los hace en función del conjunto de instrucciones de cada MP.

16/4/15



Aunque parecen circuitos similares, no lo son para nada equivalentes en lo funcional.

① está sincronizado con el CLK, el ② el reloj del FF no es CLK, sino la salida de la AND.

Un diseño con ①, me aseguro que el reloj es el mismo para todos; la señal de CLK llega a todos al mismo tiempo. Por lo tanto es el usado.

Lenguaje nemotécnico para facilitar la programación en assembler.

Ejemplo: sumar el registro R1 con R2.

ADD R1, R2

ADD R2, R1

El orden de los operando es el fabricante, cual es Fue del

- Para Intel, el 1º operando es el Destino

Los operandos pueden ser Registro y/o Memoria.

Hemos mostrado como se escriben las instrucciones

Instrucción Operando → van a estar que se use

Puedo usar la división como forma para desplazar un registro; dividiendo por 2, 4, 8, ... potencias de 2.

Ejemplo desplazar 3 lugares a derecha

$$158_{10} = \underbrace{10011110}_{19} \quad 2^3 = 8 \therefore 158 \div 8 = 19$$

$$19 = 00010011$$

Para desplazar a izquierda, multiplico mas por 2^n.

La ALU se alimenta de 2 registros seleccionados.

- ¿Que hace la unidad de control? Es quien determina cuales operandos se van a usar, que operación se va a realizar, a donde alojará el resultado, etc.
- ¿En base a qué, la unidad de control hace todo lo anterior? En base a la instrucción que pusimos

Va a leer la ROM (el programa), toma el código de la instrucción, lo decodifica y genera todas las señales que ordenan o controlan la unidad de procesamiento. Al llegar el clock, se ejecuta la instrucción.

- La Unidad de Corrimiento es puramente combinatoria, no depende del reloj.

23/4/15

La ALU es un bloque importante dentro del Data Path, y es lo encargado de realizar las operaciones aritméticas (suma, resta, incremento, decremento) y las lógicas (AND, OR, XOR, NOT).

Cuando las operaciones aritméticas son hechas por hardware, son mucho más rápidas.

La programación o "descripción" de hardware, existen 2 formas: la comportamental y la estructural. La primera describe "como quiero que funcione el hardware" y la segunda "como está construido (el circuito) el hardware".

Para algunos, la Unidad de Corrimiento está metida dentro de la ALU.

En los µP se dan operaciones de corrimiento y rotación.

• Pablos de Control, es el patrón de "0" y "1" que debe generar la Unidad de Control para configurar todos los sistemas y así ejecutar una determinada instrucción.

Por ejemplo:

A	B	D	F	H
---	---	---	---	---

Ejemplo sumar el registro R1 con R2

ADD R1, R2

ADD R2, R1

El orden de los operandos lo determina el fabricante, cual es fuente o destino. No depende del hardware.

• Para Intel, el 1º operando es el Destino

Los operandos pueden ser Registro y/o Memoria.

Hemos mostrado como se escriben las instrucciones en lenguaje nemotécnico.

Instrucción Operandos — van a estar en función del µP que se use

La ALU se alimenta de 2 registros seleccionados.

- ¿Qué hace la unidad de control? Es quien determina cuales operandos se van a usar, que operación se va a realizar, a dónde abajar el resultado, etc.
- ¿En base a qué, la unidad de control hace todo lo anterior? En base a la instrucción que pusimos

Va a leer la ROM (el programa), toma el código de la instrucción, lo decodifica y genera todas las señales que ordenan o controlan la unidad de procesamiento. Al llegar el clock, se ejecuta la instrucción

- La Unidad de Corrimiento es puramente combinacional, no depende del reloj.

23/4/15

La ALU es un bloque importante dentro del Data Path, y es la encargada de realizar las operaciones aritméticas (suma, resta, incremento, decremento) y las lógicas (AND, OR, XOR, NOT).

Cuando las operaciones aritméticas son hechas por hardware, son mucho más rápidas.

La programación o "descripción" de hardware, existen 2 formas: lo comportamental y lo estructural. La primera describe "como quiero que funcione el hardware" y la segunda "como está construido (el circuito) el hardware".

Para algunos, la Unidad de Corrimiento está metida dentro de la ALU.

En los µP se dan operaciones de corrimiento y rotación.

- Palabra de Control, es el patrón de "0" y "1" que debe generar la Unidad de Control para configurar todos los sistemas y así ejecutar una determinada instrucción.

Por ejemplo:

A	B	D	F	H
---	---	---	---	---

Los campos representan: (es lo que tiene la palabra de control)

- A: 1º operando
- B: 2º operando
- D: destino de la operación
- F: operación
- H: corrimiento, rotación o clear.

→ Register Transfer Level

- El RTL o Lenguaje de Transferencias de Registros, constituye un conjunto de expresiones y afirmaciones con una notación simbólica para especificar las interconexiones necesarias entre los distintos componentes de un sistema. Fue utilizado antes del VHDL u otro lenguaje de descripción de hardware.

30/4/15

ARM.

- El ARM soporta 3 operandos, 2 fuente y 1 destino.

Ejemplo:

$$\text{ADD } R_1, R_2, R_3$$

↴ 2º Fuente
 ↴ 1º Fuente
 ↴ Operando destino

en RTL: $R_1 \leftarrow R_2 + R_3$

similar que en C/C++: $a = a + b$

↓ destino

Posee 16 registros R0 a R15 de 32 bits cada uno. Además el bus de datos, de 32 bits de longitud; el bus de address tiene la misma longitud, 32 bits, permitiendo dirigir 4GB de memoria.

El R15 es el "program counter" o "contador de programa"

- Posee:
- un "barrel shifter" aplicado a un solo operando de la ALU.
 - registro de multiplicación, separado de la ALU, efectúa multiplicación por hardware, reduciendo tiempo
 - incrementador de PC (Program Counter)
 - Address register

Las subrutinas se manejan con CALL y RETURN.

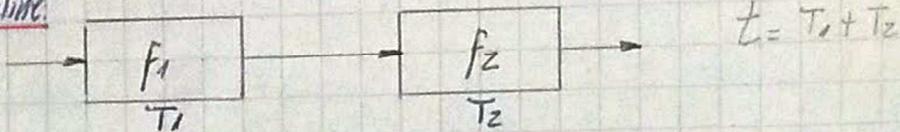
- Cuando los operandos estén en memoria, debo moverlos a registros, operar con registros y al resultado moverlo a registro o a memoria. Esto es así, debido a que las operaciones del ARM son sólo entre registros.

- Las instrucciones de procesamientos de datos, se realizan en un solo ciclo de clock.

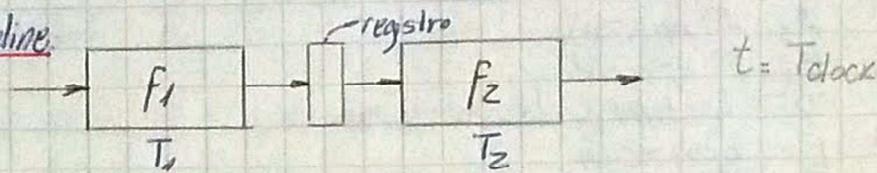
- ARM hace uso del "pipelining" (tubería o cauce) que consiste en una cadena de procesos conectados de forma tal, que la salida de cada elemento de la cadena es la entrada del próximo. Es común el uso de buffer de datos o registros entre elementos consecutivos.

Tiempo de ejecución con MP - Montaje

sin pipeline:



Con pipeline:



La ventaja del pipeline es reducir el tiempo de ejecución; pero tiene la desventaja de la latencia inicial hasta que el dato pasa de la entrada hasta lo salido.

ARM usa un pipeline de 3 etapas:

- búsqueda
- decodificación } de las instrucciones
- ejecución

→ en versiones nuevas, el pipeline es de 5 etapas, incluye las operaciones en memoria, **LOAD** y **STORAGE** (Load = cargar desde memoria, Storage = guardar en memoria)

- Con el Fetch o búsqueda de la instrucción, se busca una palabra de 32 bits, generando como resultado el código de operación (OPCODE) de la instrucción.
- Luego el bloque de la decodificación de la instrucción, es la que genera la palabra de control.
- Un operando inmediato se va a encontrar en la parte baja de la palabra del código de operación.

SUB R0, R1, #128, LSL #3 *este es lo que realiza* $R0 := R1 - 128 * 8$

↳ logical shift left 3 veces

↳ operando inmediato

desplazamiento
= logico 3 iguales

SUB R0, R1, R2, LSL#3 *realiza* $R0 := R1 - R2 * 8$

La resta siempre es en el orden (ejemplo): R1 - R2, salvo el "Reverse SUB" que efectúa: R2 - R1

14/5/15

(Usa el "power point" ARM_Teaching_Material)

• ARM es de arquitectura de 32 bits (el tamaño de su bus de datos)

• Relacionado al ARM, en su entorno:

- Byte → 8 bits
- Halfword → 16 bits = 2 Bytes
- Word → 32 bits = 4 Bytes

La mayoría de los ARM implementan 2 sets o conjuntos de instrucciones

- el "clásico" conjunto de instrucciones de 32 bits
- el "thumb" conjunto optimizado para 16 bits

Otra implementación que existe en ARM es el "Jazelle" que puede ejecutar el bytecode de Java.

Modos del Procesador.

ARM tiene 7 modos básicos de operación:

- **User** → modo "usuario", donde no se tienen privilegios (modo normal), la mayoría de los tareas comunes.
- **FIQ** → el μP entra en este modo cuando se entra/execute una interrupción de alta prioridad.
- **IRQ** → para interrupción de baja prioridad.
- **Supervisor** → es empleado en el arranque o en el reset. Sólo en la inicialización.
- **Abort** → cuando se intenta violaciones de memoria.
- **Undef** → para instrucciones no definidas.
- **System** → modo privilegiado, hace uso de los mismos registros del modo Usuario.

ARM dispone registros desde r0 a r15, pero: r13 ↔ sp(stack pointer), r14 ↔ lr(link register), r15 ↔ pc(program counter). El cpsr es el registro "estado" o de banderas. El spsr es un registro para "salvar" el cpsr cuando se cambia entre los modos.

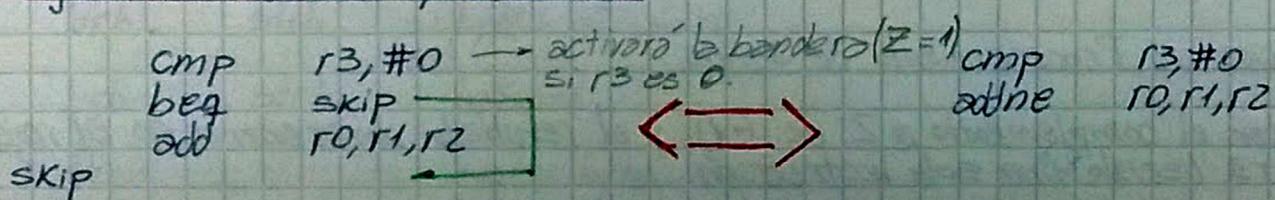
Interrupciones en ARM → Exception: excepciones

- Cuando se produce una interrupción(excepción), el ARM copia el cpsr dentro del spsr. Establece los bits apropiados en el cpsr, cambio de modo, almacena la dirección de retorno en el lr propio del modo en que cambió, y establece el PC con el vector de dirección dónde le indica la próxima instrucción.
- Para retornar de la interrupción, repone el cpsr desde el spsr (del modo en que estaba) y repone el PC desde el lr (del modo).

Program Counter (r15)

- Cuando el μP se ejecuta en estado ARM
 - las instrucciones son de 32 bits
 - todas las instrucciones deben estar alineadas en palabras (32 bits)
- Cuando el μP se ejecuta en estado Thumb
 - las instrucciones son de 16 bits
 - todas las instrucciones deben estar alineadas en halfword (16 bits)
- En estado Jazelle:
 - instrucciones de 8 bits
 - el μP lee 4 instrucciones a la vez (1 instrucción = 1 Byte = 8 bits, 32 bits = 4 Bytes = 4 instrucciones)

Ejecución Condicional y Banderas



Ambos códigos hacen lo mismo, pero uno es más reducido y se optimiza la velocidad de ejecución. (ocupa menos ciclos de máquina)

En la ALU, el shifter, opera sólo en el 2º operando.

- Por defecto, las instrucciones de procesamiento de datos no afectan a las banderas, pero opcionalmente pueden modificarse usando el sufijo 'S' en la instrucción.

loop ...

subs r1, r1, #1 r1 = r1 - 1
bne loop

sub → hace la resta pero no afecta las banderas.

subs → hace la resta y afecta las banderas.

Ejemplos de Ejecuciones Condicionales:

C

```
if(r0==0)
{
    r1=r1+1;
}
else
{
    r2=r2+1;
}
```

Assembler ARM

incondicional	Condicional
cmp r0, #0	cmp r0, #0
bne else	addeq r1, r1, #1
add r1, r1, #1	addne r2, r2, #1
b end	...
else add r2, r2, #1	...
end	...
...	...
= instrucciones	3 instrucciones
5 palabras	3 palabras
5/6 ciclos	3 ciclos

Instrucciones de Procesamiento de Datos.

→ Aritméticas: add adc sub sbc rsb rsc

→ Lógicas: and orr eor bic

→ Comparación: cmp cmn tst teq

→ Movimiento de datos: mov mvn

• Estas instrucciones trabajan sólo sobre registros, no en memoria.

28/5/15 (Continua con lo filmado "APM_Teaching_Material" disp. nº 9 en youtube)

Ejercicios de Procesamiento de Datos.

1- Hacer el complemento a 2 de (-1), y el resultado almacenarlo en el registro r3 usando una sola instrucción.

movn r3, #0

$$\begin{aligned} \neg \text{Not}(-1) &= \text{Not}(0000\ 0000) \\ &\rightarrow (1111\ 1111) \end{aligned}$$

$$\begin{aligned} (-1) &= 0000\ 0001 \\ (-1) &= 1111\ 1110 \text{ C}_2 \end{aligned}$$

2. Implementar el valor absoluto de un registro, sólo en dos instrucciones.

		Operaciones	
add s	r0, #0		
return	r0, r0, #0	SWP R0, R0	
		$r0 = r0 - r0$	

3. Multiplicar un número por 35, garantizando la ejecución en 2 ciclos de reloj (osea 2 instrucciones).

add	r9, r8, r8, lsl#2	$r9 = r8 \times 5$
rsb	r10, r9, r9, lsl#3	$r10 = r9 \times 7$

$$r9 = r8 + (r8 \cdot 2^2)$$

$$lsl\#2 = 2^2$$

$$r9 = r8 + 4 \cdot r8 = 5 \cdot r8$$

$$r10 = (r9 \cdot 2^3) - r9$$

$$lsl\#3 = 2^3$$

$$r10 = 8 \cdot r9 - r9 = 7 \cdot r9$$

- pero $r9$ es igual a $\leq r8$ entonces:

$$r10 = 7 \cdot 5 \cdot r8 = 35 \cdot r8$$

- Caso del ejercicio

El operando inmediato no puede ser un número de 32bits, ya que las instrucciones en ARM son de 32bits de longitud.

El formato de las instrucciones de procesamiento de datos, tiene disponibles 12bits para el operando 2.

Para solventar el problema, el ensamblador hace un "truco", dependiendo el tamaño de la constante a cargar.

Si el tamaño es mayor a la representación en 8bits, la carga a la constante en memoria, con la instrucción ldr. y una dirección relativa al PC.

Transferencia de Datos, entre memoria y registros.

- La memoria del sistema debe soportar todos los tamaños de acceso.

ldr str word

ldr b str b byte

ldr h str h halfword

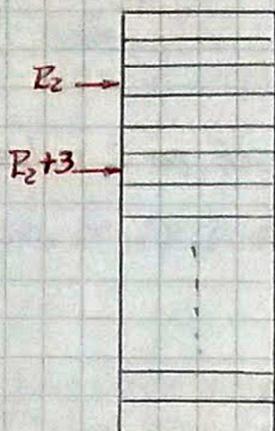
ldr sb str b signed byte

ldr sh str h signed halfword

Con ldr es cargar el contenido de la memoria a un registro.
Con str es cargar un registro a la memoria.

Especificación de la memoria.

Las direcciones accedidas por el ldr y el str es especificada por un "registro base" y un "offset".



R_2 sería mi "registro base" (un puntero)

"3" es mi offset, es una constante que se le suma a mi registro base.

El operando memoria se especificó entre corchetes.

ldr r1, [r2]

es análogo a punteros en C.

ldr r1, [r2, #3]

Offset = "desplazamiento"

El registro base es de 32 bits, el offset por valor inmediato sin signo son 12 bits ($0 \sim 4095 = 4096$ valores). El offset también puede ser un registro con desplazamiento.

ldr r0, [r1, r2]

r1 es el base

ldr r0, [r1, r2, ls1#2]

r2 es el offset

→ base

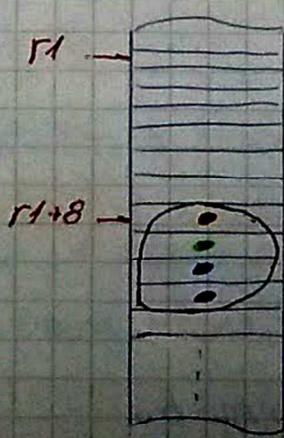
4/6/15 (continua en filmina APM - Tegorodne - Material) disp. x=18 en videoante

ldr r0, [r1, #8] → carga de la memoria a r0

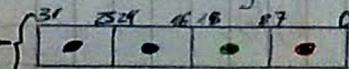
str r0, [] → almacena de r0 a la memoria

El corchete especifica "que dirección de memoria hace referencia".
Esa especificación está compuesta de dos partes:

- el registro base
- el offset



Como los registros, r_x , son de 32 bits, en memoria se organizan



A lo operación puedo agregarle para especificar b que voy a guardar, ya sea Byte, Halfword

ldr r0, [r1, #8]
-byte

FECHA
El offset puede ser sumado o restado del registro base.

Puede usarse direccionamiento pre-indexado o post-indexado.

- Pre-indexado $ldr r0, [r1, #8]!$, lo que hace es al $r1$ (base) le suma el $\#8$, el resultado lo guarda en $r0$ (actualizo el base)

- Post-indexado $ldr r0, [r1] #8$, a $r1$ (base) le suma el offset, busca el dato en la memoria y luego actualiza el $r1$ (base)

En la forma "normal" de indexado de memoria, el registro base no se altera quedo fijo.

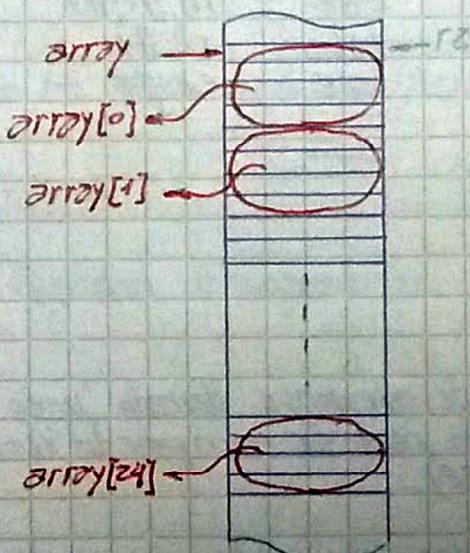
$ldr r0, [r1, #8]$

El pre y post indexado es útil cuando se usan bucles para cargar de o hacia la memoria.

Ejercicio de load/store en memoria:

Suponer un arreglo de 25 words. Un compilador asocia "y" con $r1$. La dirección base está en $r2$. Traducir la siguiente sentencia en C usando 3 instrucciones.

$array[10] = array[5] + y;$



En C, el nombre del arreglo, en este caso "array", es el puntero que apunta al primer elemento del arreglo, $array[0]$.

ldr	$r0, [r2, #5]$	$\rightarrow r0 = array[5]$
add	$r0, r0, r1$	$\rightarrow r0 = r0 + r1 \rightarrow array[5] + y$
str	$r0, [r2, #10]$	$\rightarrow array[10] = r0$

Ejercicio: hacer un programa que borre 5 Bytes en la memoria a partir de la dirección 0x2000h.

	mov	r0, #0x2000	base
	mov	r1, #0	contador
	mov	r2, #0	"borrador"
Uno	strb	r2, [r0] #1	carga en memoria, base incrementada en 1 el base
	add	r1, #1	incremento el contador
	cmp	r1, #5	compara r1 con 5
	bne	Uno	salto si r1 ≠ 5 a "Uno"
	====		

El primero que hice:

	mov	r0, #0x2000	base
	mov	r1, #0	contador/offset
	mov	r2, #0	"borrador"
Uno	cmp	r1, #5	
	bne	si	
	b	salir	
Si	strb	r2, [r0, r1]	
	add	r1, r1, #1	
	b	uno	
salir	====		

Contras:

- Uso de saltos (2 saltos seguidos)
- Modo de dirección común
- Comparación al principio

Otras soluciones propuestas: (compañeros)

①	mov	r0, #0	"borrador"	②
while	mov	r1, #0x2000	base	
	strb	r0, [r1] #1		
	cmp	r1, #0x2005		
	bne	while		
	====			

seguir	mov	r0, #0	"borrador"
	mov	r1, #0x2000	base
	mov	r2, #5	contador (decremental)
	strb	r0, [r1] #1	
	subs	r2, #1	
	bne	seguir	
	====		

Procesador Intel 8088

Los µP Intel, hasta el día de hoy tienen un modo de arranque llamado "modo real" y luego comutan al "modo protegido". El modo real se comporta igual al 8088 (un i7 en modo real se comporta como 8088).

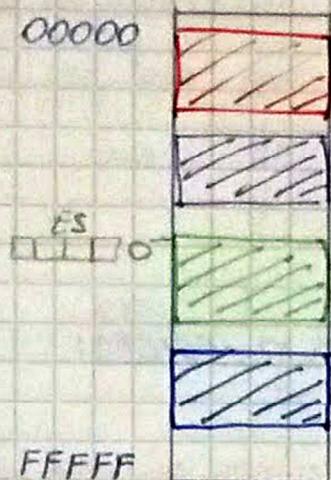
En modo protegido se habilitan ciertas funciones, que antes se implementaban por software, ahora se implementan por hardware, aumentando la performance (se parallelizan las tareas).

El mapa de memoria es único y el acceso de la memoria es segmentado. Son 4 segmentos y los denominamos:

- Segmento de Código CS
- Segmento de Pila SP

- Segmento de Datos DS
- Segmento Extra ES

El μP tiene una capacidad de direccionamiento de 1MB ($2^{20} = 1MB$).



Los segmentos pueden estar:

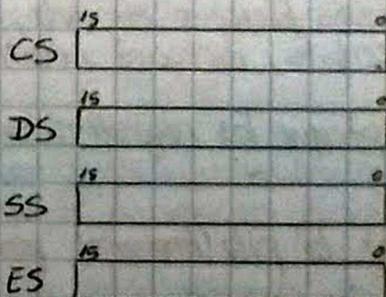
- totalmente separados
- parcialmente superpuestos
- totalmente superpuestos

La filosofía de la segmentación es para evitar borrar todo la memoria. Si borro el segmento de datos, borro sólo el de datos, no el de código por ejemplo.

- Cada segmento tiene **64KB**, en total 256KB ya que son 4 en total.
- Quien define donde está cada segmento en el mapa de memoria, son los registros (que tienen igual nombre que los segmentos) CS, DS, SS y ES. Se emplean como punteros.
Tienen un tamaño de 16bits e indican una dirección en el mapa de 20 bits.
- "Los registros de segmento, indican el comienzo de cada segmento apuntando a los 16bits más significativos de la dirección."

11/6/15

Registros de segmentos



Como mencionar a los bits:
Big Endian 0 ~ 15
Little Endian 15 ~ 0

Además de los registros de segmentos existen también los "registros generales", al igual que los anteriores, de 16bits.

Intel los denomina con letras, basados en su significado/función.

↳ En modo real: Ax, Bx, Cx, Dx

↳ En modo protegido: cuando el μP trabaja en 32/64 bits, se denominan registros "E" (Extended): EAx, EBx, ECx

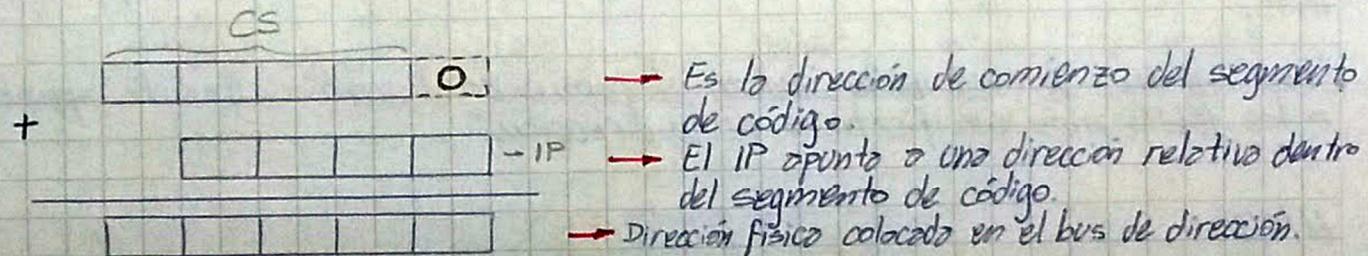
En modo real, los registros se pueden acceder como registros de 8 bits, siendo un registro de 16bits, se les puede acceder como 2 reg. de 8bits, siendo uno la parte "alta" y otro la parte "baja".

Ax	<table border="1"><tr><td>AH</td><td>AL</td></tr></table>	AH	AL	Accumulator
AH	AL			
Bx	<table border="1"><tr><td>BH</td><td>BL</td></tr></table>	BH	BL	Base
BH	BL			
Cx	<table border="1"><tr><td>CH</td><td>CL</td></tr></table>	CH	CL	Counter
CH	CL			
Dx	<table border="1"><tr><td>DH</td><td>DL</td></tr></table>	DH	DL	Data
DH	DL			

Se tiene también el conjunto de registros punteros o índices:

BP	<table border="1"><tr><td></td><td></td></tr></table>			Base Pointer
SP	<table border="1"><tr><td></td><td></td></tr></table>			Stack Pointer
SI	<table border="1"><tr><td></td><td></td></tr></table>			Source Index
DI	<table border="1"><tr><td></td><td></td></tr></table>			Destination Index
IP	<table border="1"><tr><td></td><td></td></tr></table>			Instruction Pointer

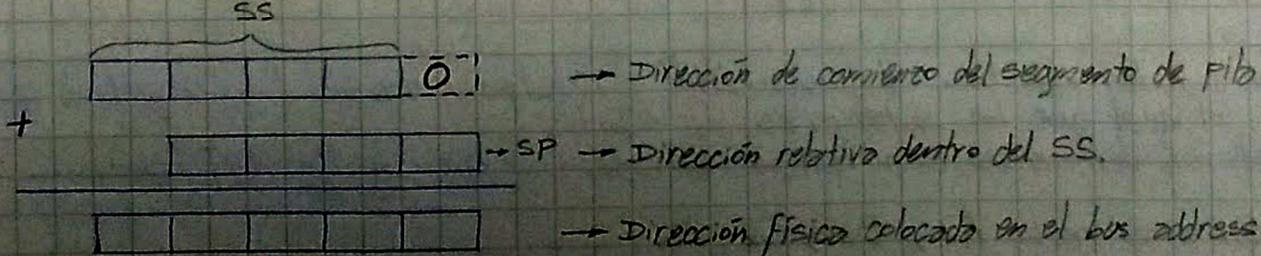
¿Cómo ubica la instrucción el MP en el segmento de código?



En Intel se denominan: dirección física o absoluta, es una dirección respecto al mapa de memoria, en este caso 20bits; dirección relativa, es una dirección de 16bits.

"Los segmentos son de 64K, porque los registros son de 16bits, $2^{16} = 64K$ ".

¿Cómo el MP ubica/direcciona la pila (memoria) para guardar el dato?



Modos de direccionamiento de Datos: no existe un "Data Pointer" análogo a los casos anteriores ya que hay varios formas de almacenar los datos.

Intel usa "mov" para registro o memoria indistintamente. (manejó 2 operandos por instrucción)

mov AL, BL
 ↴ fuente
 ↴ destino

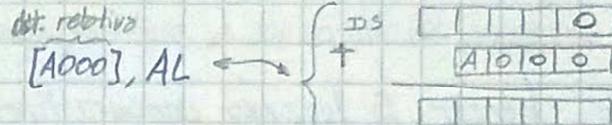
mov [], AL
 ↴ fuente
 ↴ memoria

¿Cuáles son los modos posibles?

1. Inmediato → mov AL, 7

2. Registro → mov AH, BL

3. Memoria Directa → mov [A000], AL



4. Memoria Indirecta → tiene varios modos

2. por registro base mov [Bx], AL o mov AL, [Bx]

2. por registro indice

4. por registro base + indice registro + registro

2. por registro base + desplazamiento

2. por registro indice + desplazamiento

4. por registro base + indice + desplazamiento

16 modos posibles

Las variables globales van en el Data Segment.

Las variables locales van en el Stack Segment (Pil)

En Intel para llamar a una función o subrutina **call**, para regresar de ello el programa prin

El Base Pointer (BP) actúa sólo para la Pil

Intel implementó el **byte de prefijo**, es un byte que precede a la instrucción, es un byte extra a la codificación de la instrucción. Le indica sobre qué segmento operar explícitamente.

Los modos de direccionamiento usan por defecto el Data Segment.

DS: mov [BP], AL
 ↴ byte de prefijo

Modos de direccionamiento de Datos: no existe un "Data Pointer" análogo a los casos anteriores ya que hay varios formas de almacenar los datos.

Intel usa "mov" para registro o memoria indistintamente: (manej 2 operando por instrucción)

mov AL, BL
 ↴ fuente
 ↴ destino

mov [], AL
 ↴ fuente
 ↴ memoria

¿Cuáles son los modos posibles?

1- Inmediato → mov AL, 7

2- Registro → mov AH, BL

3- Memoria Directa → mov [A000], AL

4- Memoria Indirecta → tiene varios modos

2 ↴ por registro base mov [Bx], AL o mov AL, [Bx]

2 ↴ por registro indice

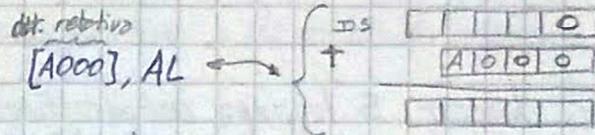
4 ↴ por registro base + indice registro + registro

2 ↴ por registro base + desplazamiento

2 ↴ por registro indice + desplazamiento

4 ↴ por registro base + indice + desplazamiento

16 modos posibles



En Intel para llamar a una función o subrutina se usa la instrucción **call**, para regresar de ello al programa principal se usa **ret (return)**

El Base Pointer (BP) actúa sólo para la pila.

Intel implementó el byte de prefijo, es un byte que precede a la instrucción, es un byte extra a la codificación de la instrucción. Le indica sobre qué segmento operar explícitamente. Los modos de direccionamiento usan por defecto el Data Segment.

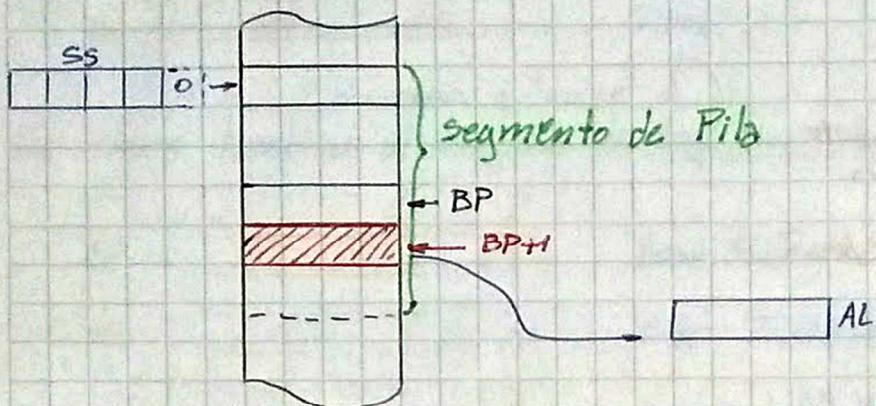
DS: mov [BP], AL
 ↴ byte de prefijo

Esto me permite "recuperar" los 16 modos de direccionamiento del Data Segment, también puede tener los 16 modos para cualquier otro segmento.

18/6/15

• Ejemplo de Byte de Prefijo:

ADD AL, [BP+1] segmento de Pila
suma el dato de la memoria (apuntado por BP+1) con el dato de AL, y el resultado lo guarda en AL



• Ejemplo: borrar 5 lugares consecutivos en la memoria (en el Data Segment) a partir de la dirección A000.

- Fuerza Bruta.

```
mov [A000], 0  
mov [A001], 0  
:  
mov [A004], 0
```

Si bien las anteriores líneas de código conceptualmente son válidas, al momento de compilarlas, el ensamblador va a dar "error", debido al tamaño del segundo operando '0', que puede ser representado en 8 o 16 bits.
El operando destino (memoria) también puede ser de 8 o 16 bits. Entonces no puede determinar el tamaño correcto.

En el ejemplo anterior, ADD AL, [BP+1], la memoria será de 8 o 16 bits, pero el operando destino es el registro AL (parte baja de Ax) que es de 8 bits de longitud, se dice que lo determina en forma implícita.

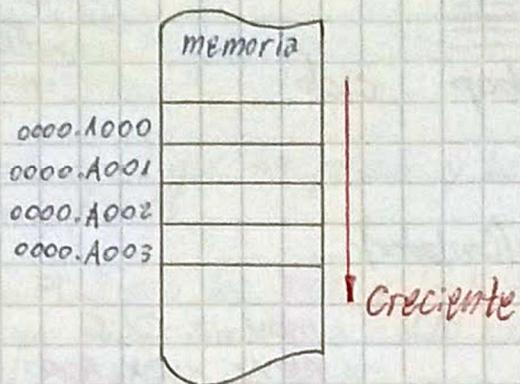
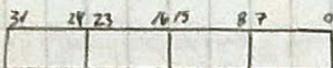
Para solucionar el problema se debe agregar "algo" a la sintaxis para explicitar el tamaño con que se trabaja.

BYTE PTR o WORD PTR
 ↳ 8 bits ↳ 16 bits

MOV BYTE PTR [A000], 0

¿Cómo se almacenan los Bytes de un registro en memoria?

Por ejemplo en ARM, un registro de 32bits lo representa como:



Intel tiene el criterio:

- byte menos significativo, dirección de memoria menos significativa
- byte más significativo, dirección de memoria más significativa

ARM hace lo contrario, byte más significativo a dirección de memoria menos significativa.

• Ejemplo:

mov [Bx], Ax

AH	AL
03	5A

se pasa a memoria

DS [] [] [] 0 →



En un lenguaje de programación como C, presenta las siguientes estructuras para resolver cualquier situación:

- Secuencia
- Selección
- Repetición

Secuencia: las instrucciones se ejecutan una a continuación de otra.

Selección: selección simple → if

selección doble → if, else

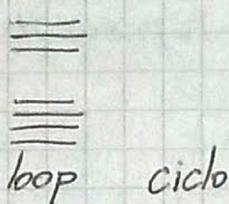
selección múltiple → switch, case

Repetición: for, while, do while

Intel implementó en su set de instrucciones la instrucción loop (bucle) para hacer la repetición.

Este instrucción emplea el registro Cx (Counter), por lo tanto habrá que asegurarse de inicializar a dicho registro con el valor requerido. Luego **bop**, decremente a Cx, verifica si llegó a cero por medio de las banderas, si no es cero se repite, sino sale.

cicb:



- Intel implementó el bucle en una sola instrucción.

- ARM requirió 3 instrucciones (2 en el mejor caso) para hacer un bucle.

- Por Bucle (Contador):

	mov	CX, 5	Carga el contador con 5
ciclo	mov	BX, A000	carga el BX con la dirección A000
	mov	BYTE PTR [BX], 0	"borra" la memoria apuntada por BX
	add	BX, 1	incremento BX en 1
	loop	ciclo	decremento Cx, si llegó a cero (flag)?, sigue o no

En este caso se usa BX porque la memoria a borrar es en el Data Segment.

Como mejora del programa sería reemplazar la linea add BX, 1 por inc BX.

Intel dispone de instrucciones de incremento y decremento:

inc
dec

- Muchas veces ARM o Intel (u otros µP) no disponen de instrucciones específicas pero pueden implementarse de otras maneras.
Ejemplo: hacer un "clear" (borrar) un registro, osea ponerlo a cero.

$$= \begin{pmatrix} \text{mov} & r1, \#0 \\ \text{xor} & r1, r1 \end{pmatrix} =$$

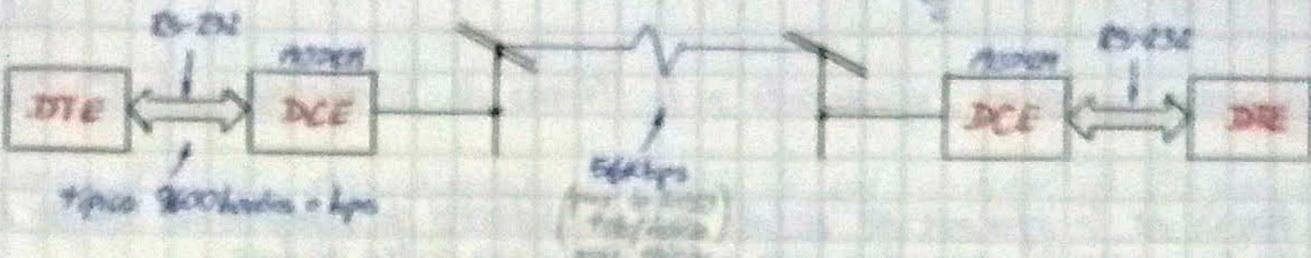
30/7/15

Puerto Serie.

Está basado en la norma RS-232 que nació en la década del '60, pensada para la comunicación a distancia por medio de la línea telefónica entre dos equipos. Estos equipos hoy en día diríamos que son los PCs, pero puede ser cualquier otra cosa.

La norma los denomina: DTE: Data Terminal Equipment → PC

DCE: Data Communication Equipment → MODEM



Este esquema representa el concepto en el que se pensó y bocetó lo que es RS-232. Es una comunicación Puerto a Puerto.

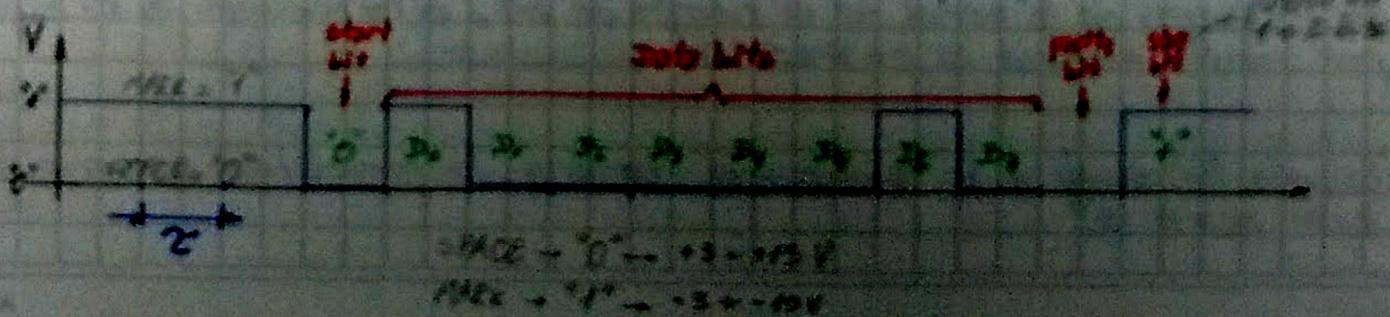
- Es posible comunicar dos DTE (dos PCs) sin los DCE y la línea telefónica, mediante el uso de un cable llamado **Null-Modem** o cable cruzado. Se hace "creer" a cada PC el escenario del esquema anterior.
- La comunicación serie se puede clasificar en 3 tipos.
 - Simplex: comunicación unidireccional.
 - Half Duplex: comunicación bidireccional pero no simultánea.
 - Full Duplex: comunicación bidireccional y simultánea.

RS-232 → Full Duplex

Bajo esta condición, mínimamente se necesitan 3 cables para establecer la comunicación, estos son:

TxD, Rxn Gnd

- Para establecer la comunicación, transmisor y receptor se deben poner de acuerdo en:
 - Tiempo de duración de bit (time bit)
 - Estado de No Transmisión (valor de '1')
 - Serial de comienzo de la transmisión o comunicación, en un estado opuesto al de No Transmisión. (start bit)
 - Cantidad de bits de datos.
 - Bit de Paridad (parity bit)
 - Estado de Fin de Transmisión (stop bit)



T : bit time (tiempo de bit)

Tipico 9600 bps

$$T = \frac{1}{V} \quad \therefore N = \frac{1}{T} \quad \text{Velocidad de Transmisión}$$

El esquema anterior representa a la Trama de Transmisión de RS-232

- La teoría de la información detalla que la Velocidad de Transmisión es en baudios o bits por segundo, pero hay una sutil diferencia entre ambos. Se considera a un bit que aporta información a aquel que de ante mano no se sabe su estado, una vez que se lo recibe se sabe si es "0" o "1". Los bits de arranque y de parada ("0" y "1" respectivamente) conocemos sus estados de antemano, por lo cual no aportan información.

Entonces, aquellos bits que no aportan información se los llama **binitis**, y los que si aportan son los **bits**.

Por lo tanto, los baudios son:

$$8 \text{ bps} = 10 \text{ baudios}$$

$$\text{baudios} = \text{bits} + \text{binitis}$$

Pero hoy en día se asume que $\boxed{\text{baudios} = \text{bps}}$

- El bit de paridad (parity bit) indica si el número de bits con valor de "1" en el conjunto de bits de datos es par o impar.

Paridad Par: el bit de paridad valdrá "1" o "0" para hacer que la cantidad de "1" transmitidos sea par.

Paridad Impar: el bit de paridad valdrá "1" o "0" para hacer que la cantidad de "1" transmitidos sea impar.

Ejemplo:

Bits de Datos	Paridad	
	Par	Impar
0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 1
1 0 1 0 0 0 1	1 0 1 0 0 0 1 1	1 0 1 0 0 0 1 0
1 1 0 1 0 0 1	1 1 0 1 0 0 1 0	1 1 0 1 0 0 1 1
1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 0

Este bit permite detectar errores en la transmisión, pero no los corrige. Si la situación lo requiere, se debe reenviar el dato.

Está muy relacionado con la probabilidad de ocurrencia de que un bit de dato cambie (por ruido o interferencia); pero que se dé 2 o más bits, la probabilidad es muy baja.

- Otra forma de evitar y/o minimizar errores de Trama, es implementar sucesivas muestras de la señal en el tiempo de bit, determinando si es "0" o "1" por simple mayoría de las muestras.

"Cuando la velocidad de transmisión aumenta, la probabilidad de error también aumenta."

6/8/15

Conector RS232.

La norma RS232, originalmente disponía de un conector de 25 pinos, que luego derivó en la versión de 9 pinos.

RS232 : 3 señales mínimas, + 6 señales de Handshaking, = 9 pines / señales (FULL) → 2 canales RS232

Estas son:

TxD: Transmitted Data →

RxD: Received Data ←

Gnd: Ground —

DTR: Data Terminal Ready —

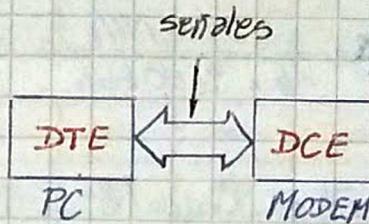
DSR: Data Set Ready —

RTS: Request To Send →

CTS: Clear To Send ←

DCD: Carrier Detect —

RI: Ring Indicator ←



el sentido de las flechas es como en el diagrama, PC — MODEM

La forma en que operan las señales de handshaking es análogo a la situación cotidiana de querer hacer una llamada telefónica.

Si se desea conectar:

- DTE y DCE → cable 1 a 1.
- DTE y DTE → cable cruzado o "Null modem"

Los señales de handshaking ("apretón de manos"), sirven para establecer un vínculo o normas para establecer la comunicación entre 2 equipos. Una vez establecidas las normas o parámetros, la comunicación se efectúa.

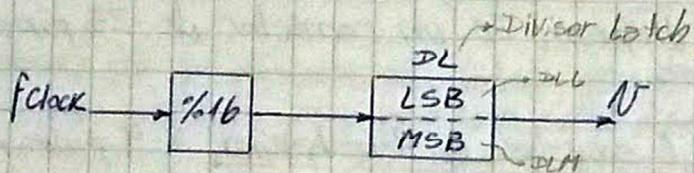
Chip de Comunicación, 16550 UART

Contiene 11 registros, mezclados en 8 direcciones consecutivas. Algunos registros comparten la misma dirección, pero se diferencian por un bit o por operación de lectura/Escritura.

divisor latch enable

El bit 7 del Registro LCR me permite acceder o habilitar los registros del "Latch Divisor", en su parte baja y alta (DLL y DLM respectivamente)

- Lo primero que se debe configurar son los parámetros de velocidad, cantidad de bits de datos, paridad, qué tipo de paridad, bits de parada. Para hacer ésto se configuran los registros: DL - LSB y MSB



$$N = \frac{f_{clock}}{16 \times DL}$$

↳ Divisor Latch

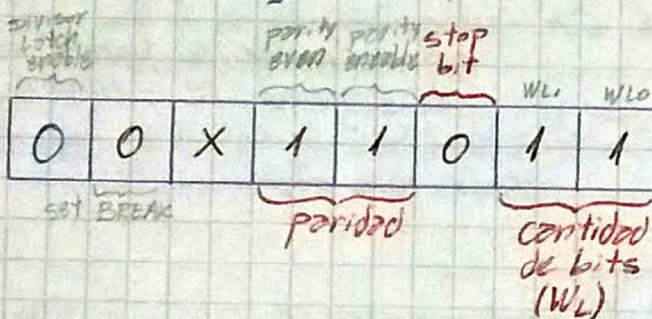
Ejemplo $N = 115200 \text{ bps}$, calculamos f_{clock} con $DL = 1$

$$f_{clock} = 1,8432 \text{ MHz}$$

¿Qué valor debo programar a DL para tener $N = 9600 \text{ bps}$?

$$DL = \frac{f_{clock}}{16 \times N} = \frac{1,8432 \text{ MHz}}{16 \times 9600 \text{ bps}} = 12_{10} \Rightarrow 000C_6 \text{ (LSB)} \quad \text{MSB} \Rightarrow \text{Este valor cargo al DL en DLL y DLM, si quiero 9600 bps}$$

• LCR: Line Control Register (Registro de Control de Línea)



Cantidad de bits: $WL_1 \quad WL_0$

0	0	--5b
0	1	--6b
1	0	--7b
1	1	--8b

stop bit: 0: 1 bit de parada
1: 2 bits de parada

paridad: → parity enable: 0: No
1: Sí

→ parity even?: 0: No
1: Sí

Par ↔ Even
Impar ↔ Odd

"Existe una regla nemotécnica para acordarse que es "odd" y "even":"

• Odd tiene 3 letras; 3 es Impar

• Even tiene 4 letras; 4 es Par

REGISTER FUNCTIONAL DESCRIPTIONS

The following table delineates the assigned bit functions for the twelve ST16C450 internal registers. The assigned bit functions are more fully defined in the following paragraphs.

Table 4, ST16C450 INTERNAL REGISTERS

A2	A1	A0	Register [Default] Note *5	BIT-7	BIT-6	BIT-5	BIT-4	BIT-3	BIT-2	BIT-1	BIT-0
General Register Set											
0	0	0	RHR [XX]	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
0	0	0	THR [XX]	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
0	0	1	IER [00]	0	0	0	0	modem status interrupt	receive line status interrupt	transmit holding register	receive holding register
0	1	0	ISR [01]	0	0	0	0	INT priority bit-2	INT priority bit-1	INT priority bit-0	INT status
0	1	1	LCR [00]	divisor latch enable	set break	set parity	even parity	parity enable	stop bits	word length bit-1	word length bit-0
1	0	0	MCR [00]	0	0	0	loop back	-OP2	-OP1	-RTS	-DTR
1	0	1	LSR [60]	0	trans. empty	trans. holding empty	break interrupt	framing error	parity error	overrun error	receive data ready
1	1	0	MSR [X0]	CD	RI	DSR	CTS	delta -CD	delta -RI	delta -DSR	delta -CTS
1	1	1	SPR [FF]	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
Special Register Set: Note *3											
0	0	0	DLL [XX]	bit-7	bit-6	bit-5	bit-4	bit-3	bit-2	bit-1	bit-0
0	0	1	DLM [XX]	bit-15	bit-14	bit-13	bit-12	bit-11	bit-10	bit-9	bit-8

Note *3: The Special register set is accessible only when LCR bit-7 is set to a logic 1.

Note *5: The value represents the register's initialized HEX value. An "X" signifies a 4-bit un-initialized nibble.

BREAK es una pausa en la comunicación de modo temporal identificado por "0" durante el tiempo de una trama.

- 0: No
- 1: Si

13/8/15

La manera en que estudiamos los Registros del chip UART lo hacemos en forma **cronológica**, cómo se deben escribir y/o leer los registros para establecer la comunicación RS-232; y no de forma como lo lista la hoja de datos de acuerdo a su ubicación o direcciones.

Hasta aquí, vimos que lo primero a hacer es configurar los parámetros de la transmisión y recepción, para ello ponemos un "1" en el bit 7 del LCR lo cual nos habilita los DLL y DLM (ubicados en la misma dirección que los RHR y THR), cargamos el valor requerido según la N deseada y seña usada. Hecho ésto, volvemos al LCR y ponemos a "0" el bit 7, configurando ahora los demás bits del presente registro.

De esta manera el Tx y Rx ya están configurados para ^{en} iniciar la comunicación.

Veamos desde el lado del Rx, "¿cómo se si se envió un dato?", se debe monitorear el registro LSR bit 0; cuando el chip de comunicación recibe un dato, ese dato se almacena en el registro de recepción RHR, posteriormente el bit 0 del LSR se pone en "1".

- LSR: Line Status Register (Registro de Estado de Línea)

--	--	--	--	--	--	--

El bit 0 (receive data ready) trabaja por "interrupción"; cuando un dato ha llegado, éste se pone en "1" para alertarle al µP de que un dato se encuentra disponible para su uso o procesamiento.

Los chips UART modernos disponen de buffers de hasta 8 Bytes, los más antiguos sólo tenían 2 Bytes.

El bit 1 (overrun error) indica si hubo datos "perdidos" o no; la pérdida de datos se da cuando: llegar un 1º dato, el µP no lo usa (no lo saca del RHR) al llegar un 2º dato, éste último sobreescribe al 1º perdiéndolo.

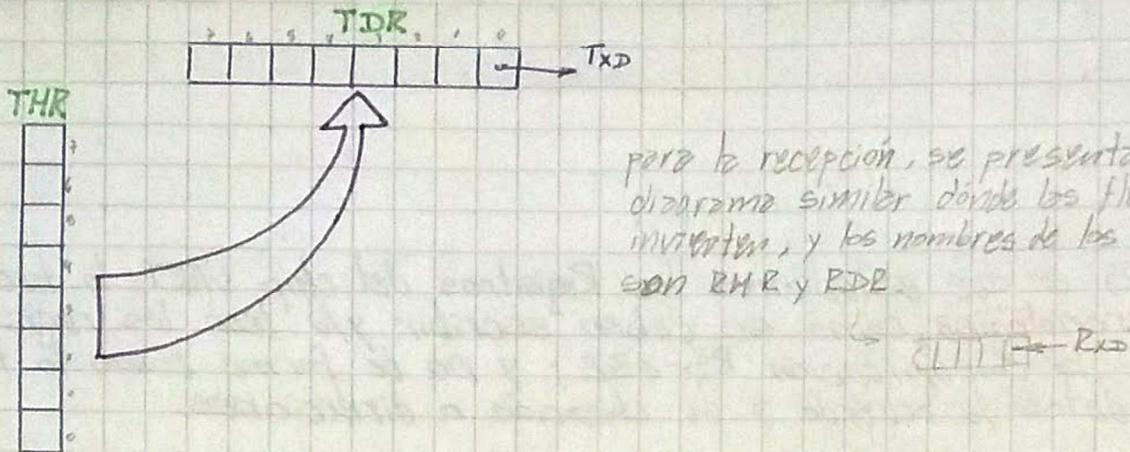
El bit 2 (parity error) indica si el dato tiene o no error de paridad, por lo tanto error en el dato recibido.

El bit 3 (framing error) indica "error de trama", por ej. Tx seteado con 8b de datos y Rx seteado con 5b de datos. Se tienen diferentes tramas, causando un error.

El bit 4 (break interrupt), se pone a "1" si en la linea se puso un break ("0") por más de un tiempo de trama.

min 35

Los bits 5 y 6 trabajan con el registro de transmisión THR y TDR, este último registro no es accesible por el programador.



para la recepción, se presenta un diagrama similar donde las flechas se invierten, y los nombres de los registros son RHR y RDR

Cuando se desea transmitir un dato, cargo el dato en el THR (Transmission Hold Register), luego el chip UART lo transfiere al TDR, siendo éste un registro de desplazamiento, donde la salida es efectivamente el pin TxD del puerto serie.

Debido a la diferencia importante entre la fcbk del chip UART y la del µP (velocidad de procesamiento), el µP se debe cerciorar de que puede escribir y no sobre escribir un dato.

Entonces los bits 5 y 6 indican si los registros THR y TDR están liberados o vacíos para ser cargados con nuevos datos a transmitir.

Si $b5=1$ y $b6=1$, no hay ningún dato pendiente de ser transmitido y que todo ya se transmitió.

min 48

Existe también un registro de control de módem (MCR) y un registro de estado de módem (MSR). En el MCR se ubican el DTR y RTS, (señales de handshaking). En el MSR, se monitorea el estado del módem con las otras señales de handshaking CD, RI, DSR y CTS. También existen los bits ΔCD , ΔRI , ΔDSR y ΔCTS .

En el MCR se encuentran los OPI y OPZ que están vinculados de forma externa, son 2 pines del chip de comunicaciones, específicamente son 2 salidas. En la PC están empleados para resetear el módem, aunque no en todas se lo implementa.

El bit 4 del MCR (loop back) tiene como particularidad de probar el estado del chip UART; cuando se setea a "1", el chip cambia internamente la Txo hacia Rxo sin salir hacia afuera por el puerto. De esta forma si se envía un dato, el dato recibido debe ser el mismo que el enviado si todo funciona correctamente.

1h 10min:

Se disponen además de registros de Interrupciones, el IER (Interrupt Enable Register) para permitir o no las interrupciones, y el ISR (Interrupt Status Register).

Conversion A/D. 1h22min

Señal Analógica: es aquella que presenta una variación continua con el tiempo. Estas señales predominan en nuestro entorno ejemplo: temperatura, presión, distancia, etc.

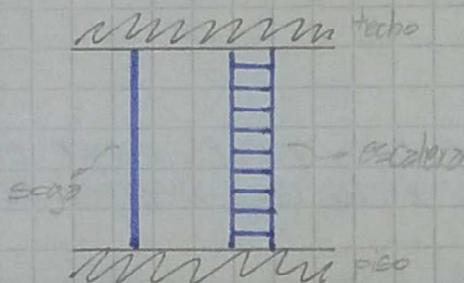
Señal Digital: es aquella que presenta una variación discontinua con el tiempo y que sólo puede tomar ciertos valores discretos.

1h39min

Error de Cuantificación: es un error inherente al proceso de cuantificación. Es posible reducirlo pero no eliminarlo.

20/8/15

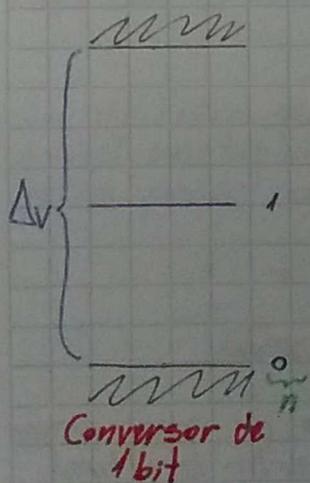
Para pensar y comprender el proceso de conversión A/D, es útil imaginar una habitación en la cual desde el techo al piso se encuentran una soga y una escalera. La soga es un medio continuo, análogo a las señales analógicas y la escalera es un medio discreto (por peldaños o escalones), análogo a los señales digitales.



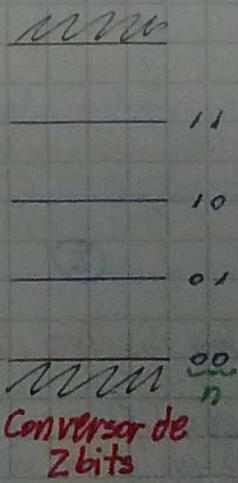
Si imaginamos un lím cuando la separación entre los peldaños de la escalera tiende a cero (sería como ir agregando escalones uno al lado del otro) el resultado obtenido sería una tabla, un medio continuo. (error igual a cero)

Basado en lo análogo, el error máximo cometido (en la escalera) es de $\pm \frac{1}{2}$ de la separación entre escalones.

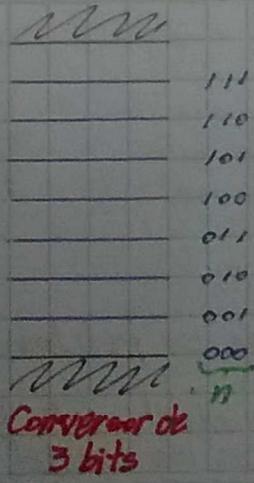
Un dato importante en los Conversores A/D es el Rango de Tensión de Entrada a Convertir (ΔV). Ejemplo: $\Delta V = 0 \sim 6V$



Conversor de 1bit



Conversor de 2bits



Conversor de 3bits

Los Conversores A/D se clasifican por el número de bits que disponen o utilizan. Ejemplos: 8 bits, 10 bits, 12 bits, etc. Entre más bits tenga, más niveles de comparación se tienen y a su vez se disminuye el error.

El tamaño, separación o rango del nivel es lo que se denomina el LSB (Less Significant Bit, Bit Menos Significativo). Como se observa en los anteriores gráficos, entre un nivel y el siguiente (o anterior), el bit que cambia es el de menor peso.

En forma general, el LSB se expresa como:

$$LSB = \frac{\Delta V}{2^n}$$

ΔV : rango de tensión de entrada

n: cantidad de bits

El error de cuantificación será entonces:

$$e_q = \pm \frac{1}{2} LSB$$

Ejemplo: ¿Cuál sería el e_q para un conversor A/D cuya $\Delta V = 0 \sim 5V$ y $n = 8$ bits?

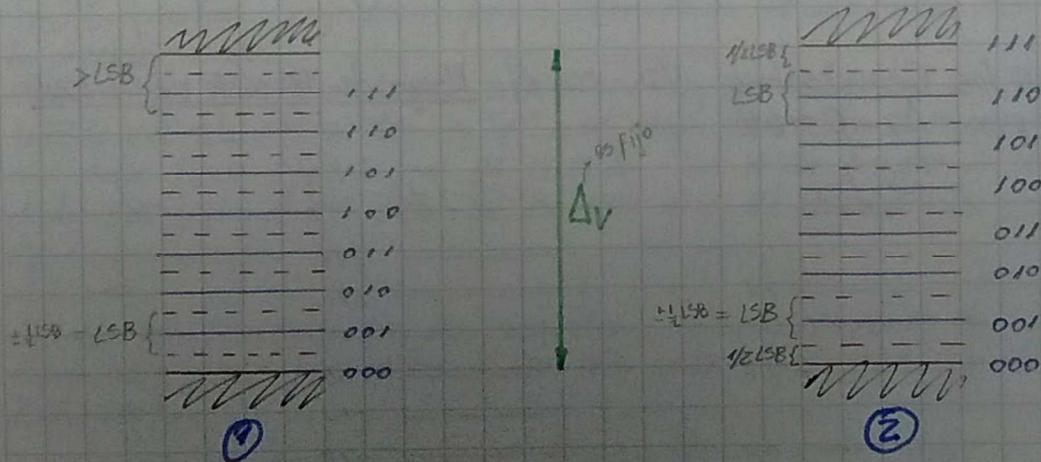
$$e_q = \pm \frac{1}{2} \cdot \frac{\Delta V}{2^n} = \pm \frac{1}{2} \cdot \frac{5V}{2^8} = \pm 9,7656 mV \approx \pm 10mV$$

min 25:

- Cuando se tiene un error producido por circuito, que es mayor al e_q del conversor A/D, entonces no se tiene verdaderamente un conversor de n bits. Los n bits no son reales.
- La alimentación del conversor A/D, también es factor influyente en el error de cuantificación. Esto es provocado por la fuente de alimentación de CC, si tiene superpuesto ruido o rizado, cuya amplitud sea comparable al valor del e_q , también no se tendrá un conversor de n bits.

min 28:

Niveles de Comparación:



--- Los trazos en puntos indican los niveles de comparación

- ① En el último nivel, esta forma de cuantificación presenta un error mayor a un LSB. Todos los demás presentan un error de $\pm \frac{1}{2} LSB$ salvo la última

cuenta que podría haber más de $\frac{1}{2}$ LSB de error.

② "Mejora" a ①, ubicando el último nivel de cuantificación en el "techo" (lo que se dio el máximo de ΔV); los niveles de cuantificación siguen siendo los mismos ya que sigue teniendo N bits, salvo que el rango o distancia entre estos es diferente. Entonces el LSB queda como:

$$LSB = \frac{\Delta V}{2^{N-1}} \text{ [Volts]}$$

por las ecuaciones de LSB, este último presenta mayor eq

min 48:

Ejemplo: Cuantos bits debería tener un conversor A/D con $\Delta V = 0 \sim 5V$ y un error máximo de $7mV$?

Es posible determinarlo por las dos maneras:

$$|Eq| = \pm \frac{1}{2} LSB = |7mV| \quad \therefore |LSB| = |14mV|$$

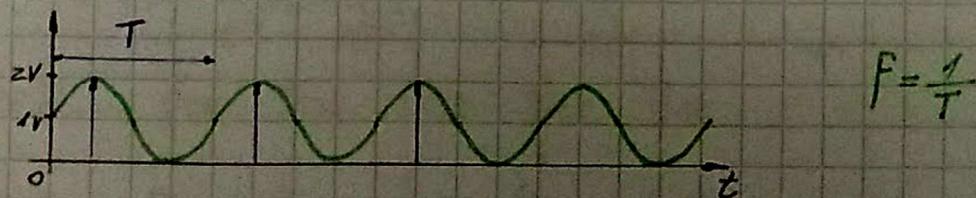
$$1) \quad LSB = \frac{\Delta V}{2^n} \quad \therefore n = \frac{\log(\frac{\Delta V}{LSB})}{\log(2)} = \frac{\log(\frac{5V}{14mV})}{\log(2)} = 8,48 \text{ bits} \approx 9 \text{ bits}$$

$$2) \quad LSB = \frac{\Delta V}{2^{n-1}} \quad \therefore n = \frac{\log(\frac{\Delta V}{LSB} + 1)}{\log(2)} = \frac{\log(\frac{5V}{14mV} + 1)}{\log(2)} = 8,48 \text{ bits} \approx 9 \text{ bits}$$

- el resultado de n siempre se redondea hacia arriba, no importa si dio $8,1$, $8,6$. Esto es debido a que redondear hacia abajo aumenta el error, pudiendo ser mayor que el error mayor máximo pedido

- La precisión en la conversión está dada por el número de bits del conversor no siendo el único factor para decidir o elegir un conversor A/D. Este otro factor es el **tiempo de conversión**, que está directamente relacionado con la señal (concretamente con su velocidad o frecuencia) a convertir.

Lo primero que hace el conversor A/D, es tomar una muestra de la señal analógica cada intervalos de tiempo T . Luego retiene esa muestra para poder convertir la muestra a un valor binario.



1h 9min.

Teorema del Muestreo: la frecuencia de muestreo debe ser al menos el doble de la máxima frecuencia contenida en la señal.

→ Frec de muestreo → muestreo

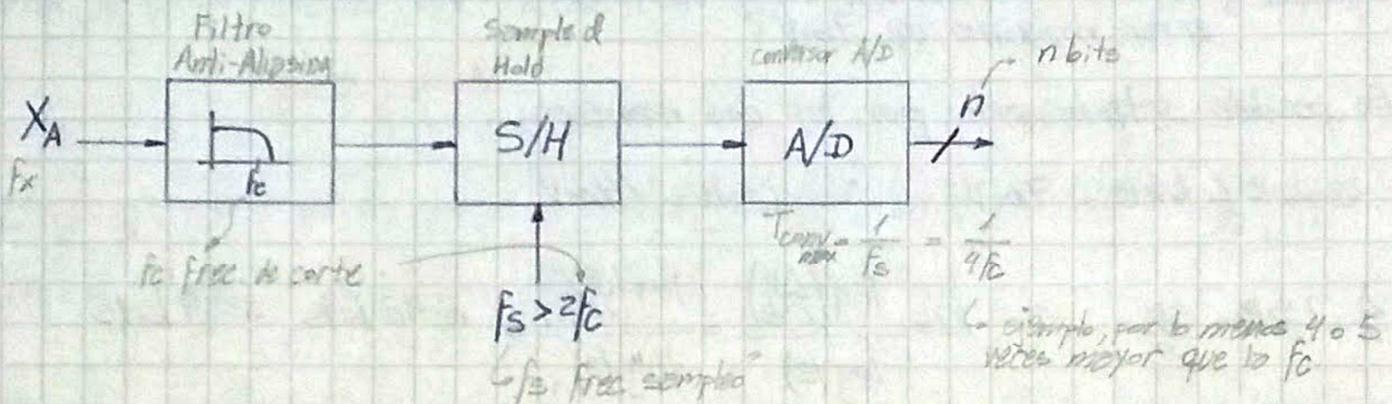
$$f_s \geq 2f_x$$

Frec. de la señal

No es restricción que la f_s sea $2f_x$, de hecho, una mayor f_s por ej: $4f_x$ o $10f_x$, obtendría una mejor aproximación de la f_x , pero nunca llegaría a la red f_x (serían infinitas muestras).

- La velocidad del conversor depende de la f_s , y f_s está dada por la f_x

Esquema del Conversor A/D



Filtro Anti-Aliasing: es un filtro paso bajos, atenuando las frecuencias altas (mayores que la frecuencia de Nyquist), con lo cual la señal a la salida del filtro se dice que está limitada en banda.

Sample & Hold: realiza la muestra de la señal a su entrada cada cierto $T_s = 1/f_s$ y retenido el tiempo necesario para ser cuantificado.

27/8/15

De querer convertir por ej 8 canales analógicos, no es práctico repetir 8 veces el esquema del conversor A/D, para solucionarlo, es posible implementar a la entrada un Multiplexor de 8 entradas a 1 salida. Esto implica que el tiempo de conversión del A/D, ahora debe ser (según el ejemplo) 8 veces menor, osea más rápido que para un solo canal.

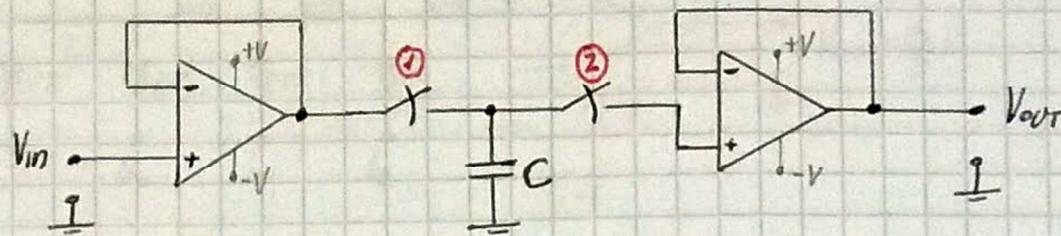
$$T_C = \frac{1}{5f_c} = \frac{1}{f_s}$$

→ es un ejemplo
podría ser mayor

$$T_{\max \text{ conversor}} = \frac{T_C}{8} = \frac{1}{40f_c}$$

→ es debido a la cantidad
de canales a convertir

Circuito Sample & Hold (S/H). min 14



Aquí se observa un circuito S/H en su idea y con el uso de A.O. Los A.O se encuentran en configuración "seguidor de tensión", presentando las siguientes características:

- $A_V = 1$ • Adopta Impedancia
- alto Z_i
- bajo Z_o

Esta configuración actúa de Buffer. Por ejemplo, la señal V_{in} será la señal obtenida desde un transductor o sensor; ciertos sensores dan señales de tensión de muy bajo amplitud (mV o µV) y no tienen capacidad de dar corriente, por lo tanto, si se lo aplica directamente al C, éste no se cargaría y el circuito no cumpliría con el "muestreo y retención" de la señal. Por este motivo, la corriente de carga del C la provee el AO en su salida por medio de su fuente de alimentación.

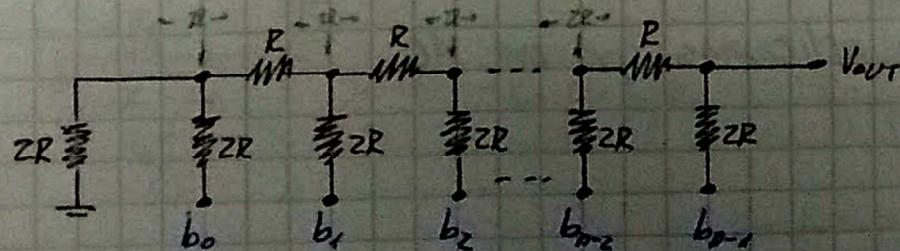
Analisis:

El proceso de muestreo o "sample" se lleva a cabo con ① cerrado y ② abierto, así la señal de V_{in} aparece a bornes de C y éste no tiene camino por donde descargarse.

Ahora, la retención o "hold" y posterior salida al conversor A/D, se hace con ① abierto y ② cerrado, la carga en C se mantiene debido a la alta impedancia (∞ idealmente) de entrada del AO, entonces $V_{out} = V_C = V_{in}$. y el ADC convertirá el valor de V_{out} . Terminada la conversión, el proceso de S/H se repite para la próxima muestra.

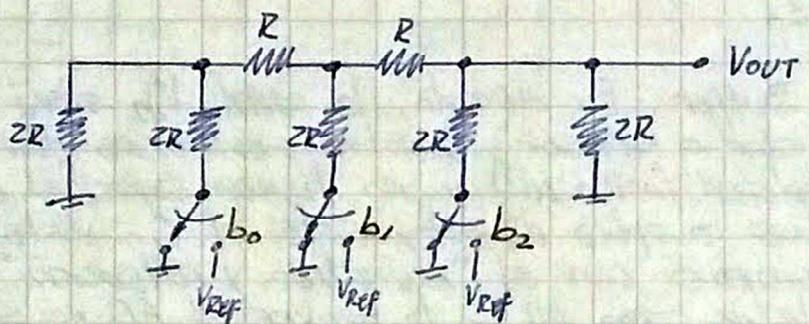
Conversor D/A. min 35

Este tipo de conversor hace el camino inverso de lo visto hasta aquí, convierte una señal digital (discreta) en una señal analógica (continua). El circuito más utilizado es el denominado R2R, formado por unos mallas de resistencias sólo de dos valores posibles: R y $2R$.



Completer cómo funcionan con los videos

Por ejemplo si fuese un conversor de 3 bits:



La tensión de salida sería la suma de los aportes de cada bit:

$$V_{OUT} = \frac{V_{REF}}{3} \cdot b_2 + \frac{V_{REF}}{6} \cdot b_1 + \frac{V_{REF}}{12} \cdot b_0$$

Sacando factor común:

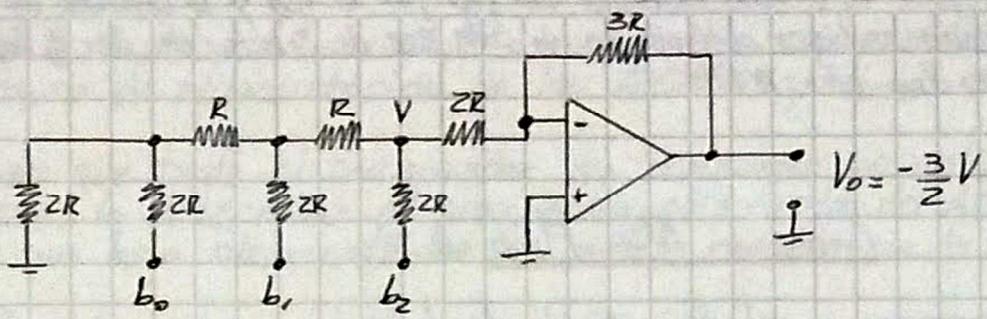
$$V_{OUT} = \frac{V_{REF}}{12} (4 \cdot b_2 + 2 \cdot b_1 + 1 \cdot b_0)$$

$$V_{OUT} = \frac{V_{REF}}{12} (Z^2 \cdot b_2 + Z^1 \cdot b_1 + Z^0 \cdot b_0)$$

En forma general, para n bits:

$$V_{OUT} = \frac{V_{REF}}{3 \cdot Z^{(n-1)}} \cdot \sum_{m=0}^{m=(n-1)} Z^m \cdot b_m \quad n: \text{cantidad de bits}$$

Si a este circuito lo aplicamos con un AO:



Así:

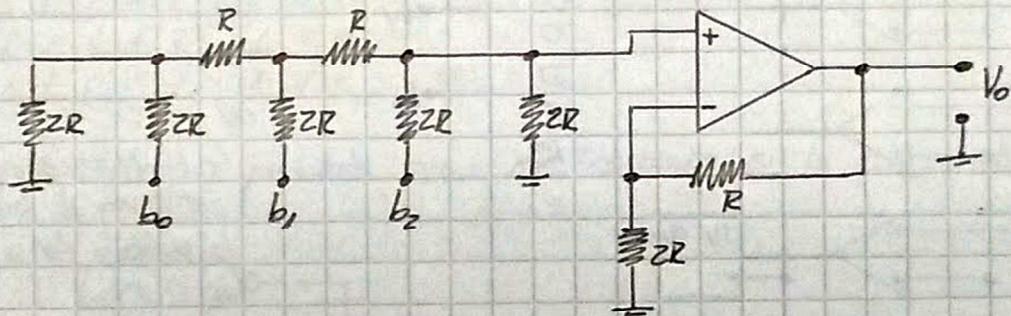
$$V_o = -\frac{V_{REF}}{Z^3} (Z^2 b_2 + Z^1 b_1 + Z^0 b_0)$$

En general:

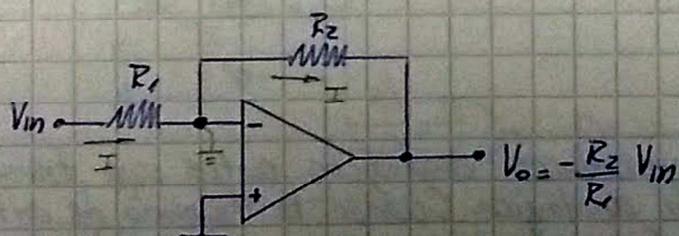
$$V_o = -\frac{V_{REF}}{Z^n} (Z^{n-1} \cdot b_{n-1} + Z^{n-2} \cdot b_{n-2} + \dots + Z^1 \cdot b_1 + Z^0 \cdot b_0)$$

El inconveniente de este circuito es que el salido aparece invertido, ya que el AO está en configuración inversora.

Para solventar ese problema, se usa la configuración no inversora.

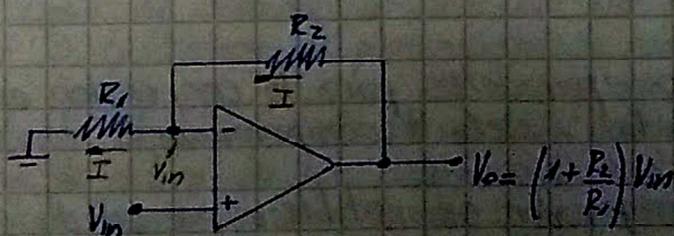


La determinación de los anteriores circuitos se basan en los siguientes:



$$I = \frac{V_{in}}{R_1} \quad I = \frac{-V_o}{R_2} \quad \therefore I = I$$

$$\frac{V_{in}}{R_1} = -\frac{V_o}{R_2} \Rightarrow V_o = -\frac{R_2}{R_1} \cdot V_{in}$$



$$I = \frac{V_o}{R_1 + R_2} \quad I = \frac{V_{in}}{R_1} \quad \therefore I = I$$

$$\frac{V_{in}}{R_1} = \frac{V_o}{R_1 + R_2} \Rightarrow V_o = \frac{R_1 + R_2}{R_1} \cdot V_{in}$$

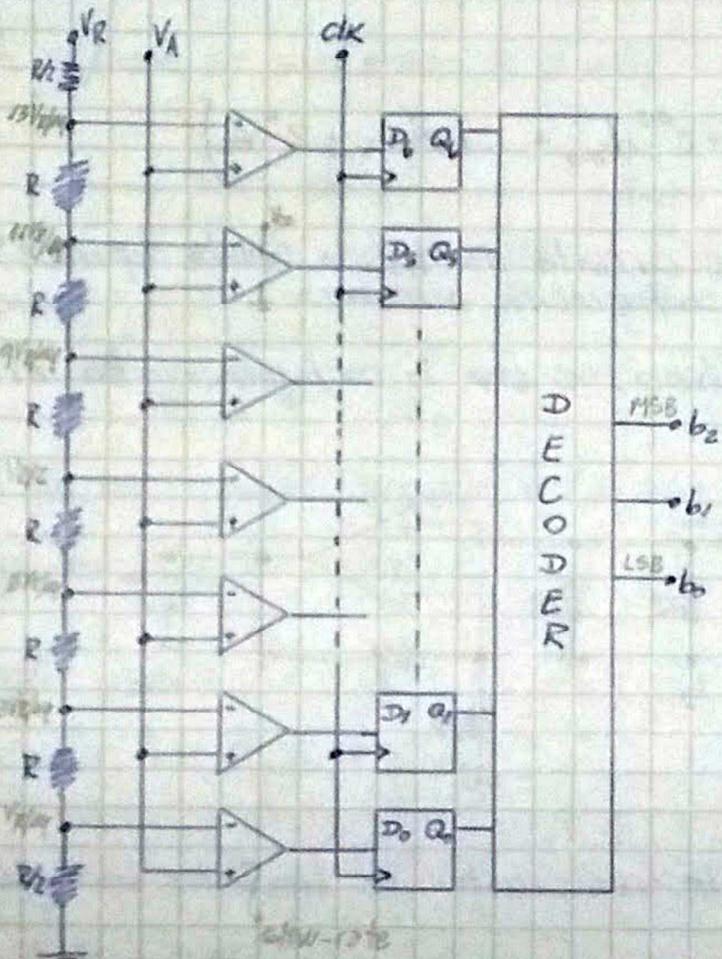
3/9/15

En un D/A el "paso" de la señal analógica se fija por la V_{ref} y la cantidad de bits, $V_{ref}/2^n$.
Ejemplo: la máxima tensión analógica obtenida en un D/A R2R de 3 bits es de $\frac{3}{4} V_{ref}$ - solo de la ecuación
Voir numero décimal V_{ref} para n bits.

Conversor A/D Flash min 27

Este tipo de conversor es el más rápido de todos los convertidores, su tiempo de conversión es el más corto. Esto se debe a que es una arquitectura en "paralelo".

Ejemplo: conversor de 3 bits.



+Vcc	11111111	111
$V_{ref}/2^3$	-	110
$V_{ref}/2^2$	-	101
$V_{ref}/2^1$	-	100
$V_{ref}/2^0$	-	011
$V_{ref}/2^{-1}$	-	010
$V_{ref}/2^{-2}$	-	001
$V_{ref}/2^{-3}$	-	000
0V	-	-

→ el techo/pico es el intervalo de entrada a cargo de entrada ΔV a convertir V_2 (de 0 a V_2)

$$V_{LSB} = \frac{V_{high} - V_{low}}{2^n}$$

Cómo se observa, se utilizan comparadores analógicos entre una tensión de referencia determinada y la tensión analógica. Cuando la tensión de entrada analógica supera la tensión de referencia de un comparador determinado, se genera un nivel alto.

En general se requieren $2^n - 1$ comparadores para la conversión de un código binario de n bits. Del ejemplo: 3 bits $\therefore 2^3 - 1 = 7$ comparadores.

La tensión de referencia de cada comparador se establece por un divisor de tensión.

Lo señal analógico V_A a convertir, está presente en la entrada No inversora de todos los A/Ds, por lo tanto todos ellos realizan la comparación al mismo tiempo (en

paralelo). La salida de los comparadores estará comprendida en el rango de la tensión de alimentación de los AO ($0V$ y $+Vcc$, $0 - Vcc$ y $+Vcc$)

El tiempo que tarda el comparador en reaccionar su salida ante cambios en su entrada, lo determina el parámetro slew-rate del AO. Se debe buscar un AO con bajo slew-rate si se desea rapidez en la comparación.

Hasta el punto de los comparadores, puede decirse que se hace el "sampleo"; falta el "hold" o retención, esto se hace con los flip-flops D controlados por un clock, cuando se active el dato presente en las entradas D_x pasa a las salidas Q_x , reteniendo estos datos hasta el próximo pulso de clock.

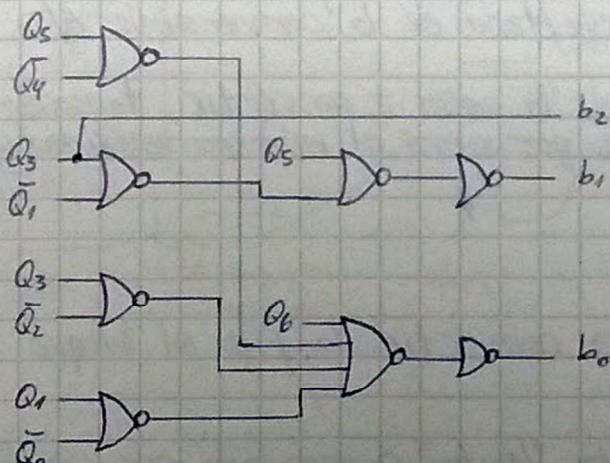
Luego las salidas de los FF, entran al bloque DECODER o circuito combinacional que transforma (para el ejemplo) 7 entradas a 3 salidas, el cual es caracterizado por una Tabla de Verdad. se vuelve complicado de reducir a menor. se usan programas
 $Z^7 = 128$ combinaciones P/ el ejemplo

La siguiente Tabla de Verdad reducida representa al bloque Decoder del ejemplo:

Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0	b_2	b_1	b_0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	1	0	1	0
0	0	0	0	1	1	1	0	1	1
0	0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1

se supone que el resto de las combinaciones no se van a dar

Una implementación posible para el Decoder es la siguiente:



En este caso el Decoder insume un tiempo máximo, desde la entrada a la salida de 3 tp (tiempo de propagación de compuerta).

Por lo tanto el tiempo de conversión está determinado principalmente por el tiempo de retardo de los comparadores (slew-rate) y el tiempo máximo de propagación existente en el Decoder.

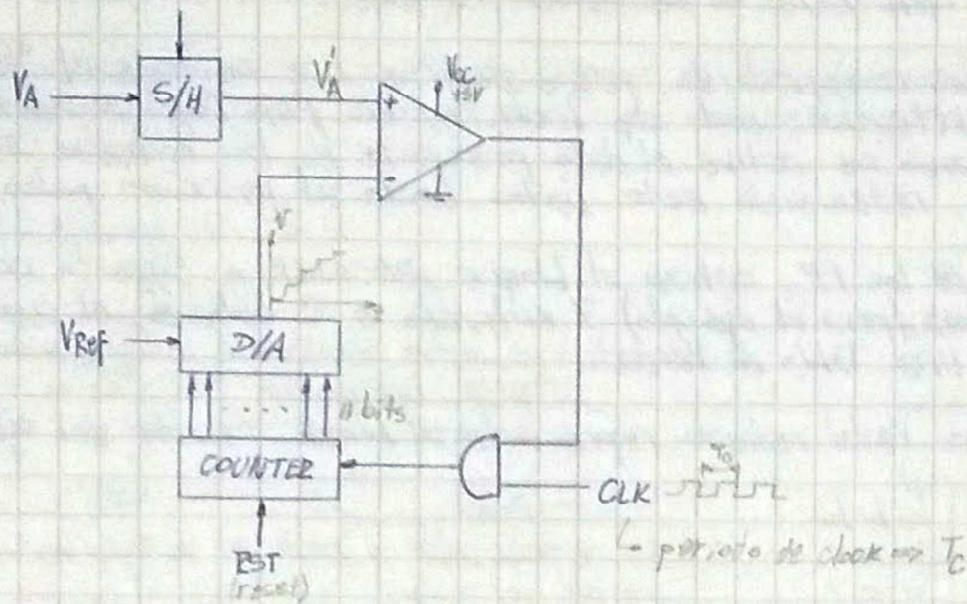
• Una gran desventaja de esta arquitectura es que cuanto mayor la cantidad de bits de salida se necesitan mayor cantidad de componentes, aumentando costo, consumo

y dissipación de calor.

10/9/15

Conversor A/D escalonado (o rampa en escalera)

Este tipo de conversor implementa un D/A.



En el momento inicial, el Counter se pone a cero con el RST; comienza a contar a medida que le lleguen los pulsos de CLK, siendo controlados o restringidos por la compuerta AND en conjunto con la salida del comparador analógico. El comparador evalúa si la señal en (-) es mayor que la señal del D/A en (-), si ésto no se da, el comparador tiene por salida un "1" (+Vcc) y el contador cuenta.

Cuando la señal del D/A supera a la señal presente en (-), la salida del comparador se hace "0" (GND), inhibiendo los pulsos de CLK, con lo cual la salida del Counter es el resultado de la conversión A/D.

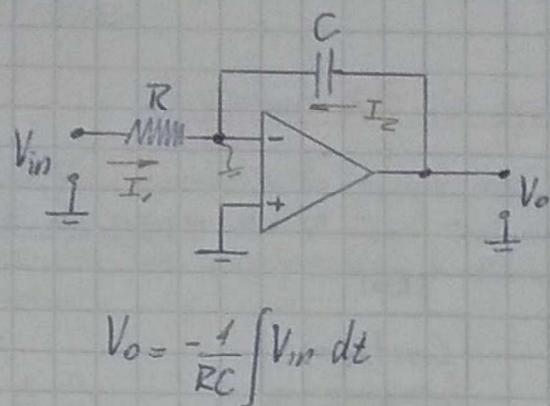
El tiempo de conversión depende de la señal a convertir, tomando como máximo el caso más desfavorable, que será el número máximo que puede alcanzar el Counter.

$$2^n \cdot T_0$$

El rango máximo de tensión a convertir lo fija el bloque D/A.

Si bien este conversor resulta "lento" comparado con un flash, resulta útil en aplicaciones donde el tiempo de conversión no sea algo crítico. Por ejemplo: un termómetro digital.

Es posible reemplazar el bloque D/A (que genera una señal discreta de tensión escalonada) por un circuito totalmente analógico que generará una rampa de tensión continua, por ejemplo: un integrador con AC.



$$I_1 = \frac{V_{in}}{R}$$

$$I_2 = \frac{CdV_o}{dt}$$

$$I_1 = -I_2 \quad : \quad \frac{V_{in} + C \frac{dV_o}{dt}}{R} \approx \text{despejo } V_o$$

(integro a ambos lados)

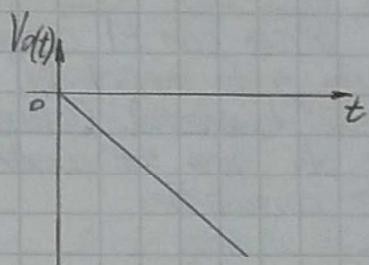
$$V_o = -\frac{1}{RC} \int V_{in} dt$$

Para obtener una rampa en V_o , la señal de entrada $V_{in} = \text{cte} = A$.

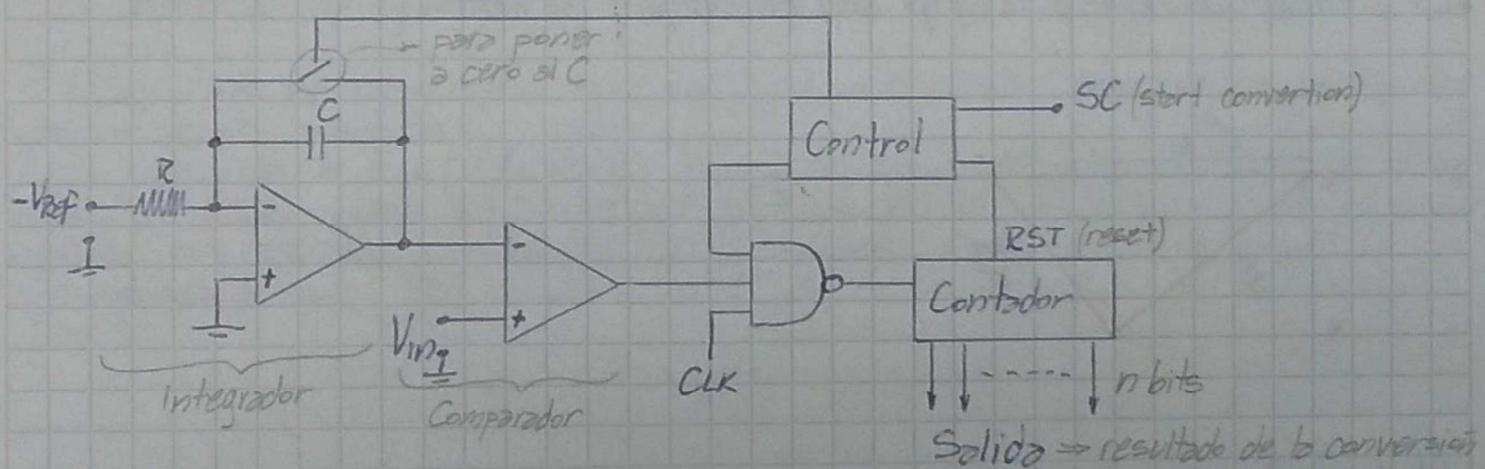
$$V_o = -\frac{1}{RC} \int V_{in}(t) dt = -\frac{1}{RC} \int A dt$$

$$V_o = \left(-\frac{A}{RC}\right)t$$

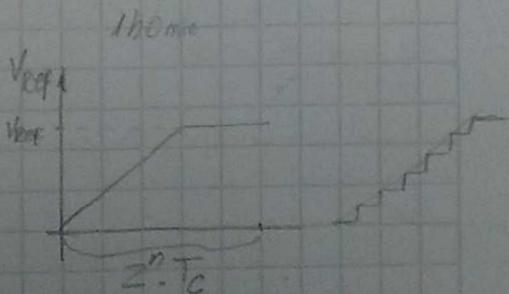
pendiente de la rampa



Conversor Simple Rampa 42 min



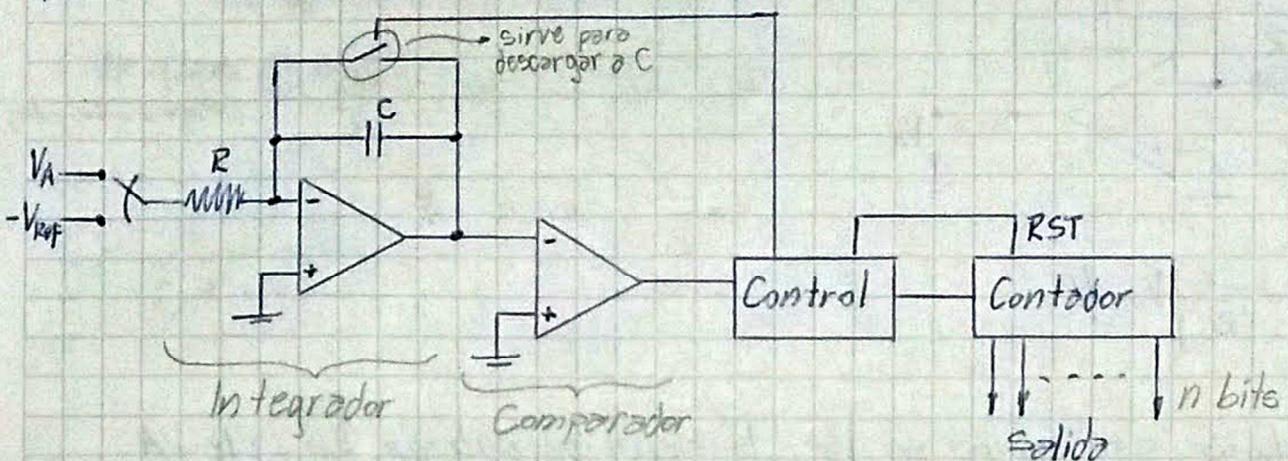
Salida \Rightarrow resultado de la conversión



el RC debe ser preciso para evitar errores
por ej: temperatura V_o(t) más grande que el CLK

Convertidor Doble Rampe. 1b

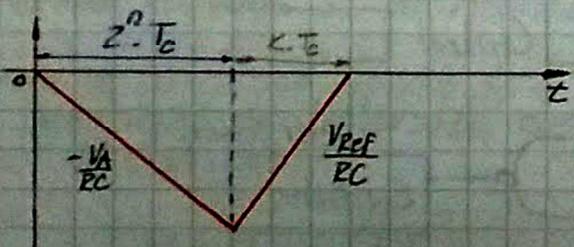
Es muy usado en multímetros digitales por ejemplo. Presenta mejor precisión que el Simple Rampe.



El circuito, inicia con el capacitor C en cero (sin carga). Inmediatamente se conecta a la señal V_A y se dejó que el Contador llegue a lo máximo cuenta ($0 \sim 2^n - 1$ vueltas). Cuando la cuenta pasa del valor máximo a 0 nuevamente, es decir que se conecta a $-V_{REF}$ y comienza el proceso de cuenta en esta condición.

Dado que los cuentos para V_A y para $-V_{REF}$, el circuito usa los mismos valores de RC , se minimizan los errores (por ej: corrimiento por temperatura) haciéndolo bastante preciso.

- La cantidad de bits, se fija con el bloques Contador.



$$Z^n \cdot T_C \cdot \left(-\frac{V_A}{RC} \right) = T_C \cdot K \cdot \frac{V_{REF}}{RC}$$

$$-V_A = K \cdot \frac{V_{REF}}{Z^n} \quad K: \text{cuenta}$$

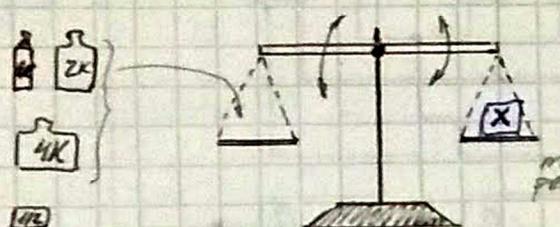
- El tiempo máximo de conversión es:

$$Z^{n+1} \cdot T_C$$

17/9/15

Conversor A/D de aproximaciones sucesivas (SAR)

Su funcionamiento puede ser asemejado a una balanza romana o balanza de brazos en la cual se tienen dos platillos; uno donde se colocan pesos de valores conocidos, y en el otro, el objeto o masa a determinar el peso.



El proceso de medición es sencilla, se coloca la masa "X" desconocida en uno de los platillos y en el otro se coloca un peso o peso patrón y se observa hacia dónde se inclina la balanza.

Dependiendo hacia qué lado se incline la balanza, nos dará una idea que objeto es más pesado que el otro, si habrá que quitar o agregar masas patrón/les, haciendo el procedimiento repetitivo hasta poder (luego de una serie de pesados) determinar la masa X.

En el caso del ejemplo de la figura, podemos decir que la balanza tendrá:

Rango: 0 ~ 8 kg

Error: $\pm 1/4 \text{ kg}$

• El Rango de la balanza es el doble del patrón más grande que se disponga. El error es $\pm 1/4 \text{ kg}$ = la mitad del patrón más chico disponible.

"El conversor de aproximaciones sucesivas opera de igual forma"

En cuestión de velocidad de conversión, es rápido comparado contra los conversores escalera y rampas.

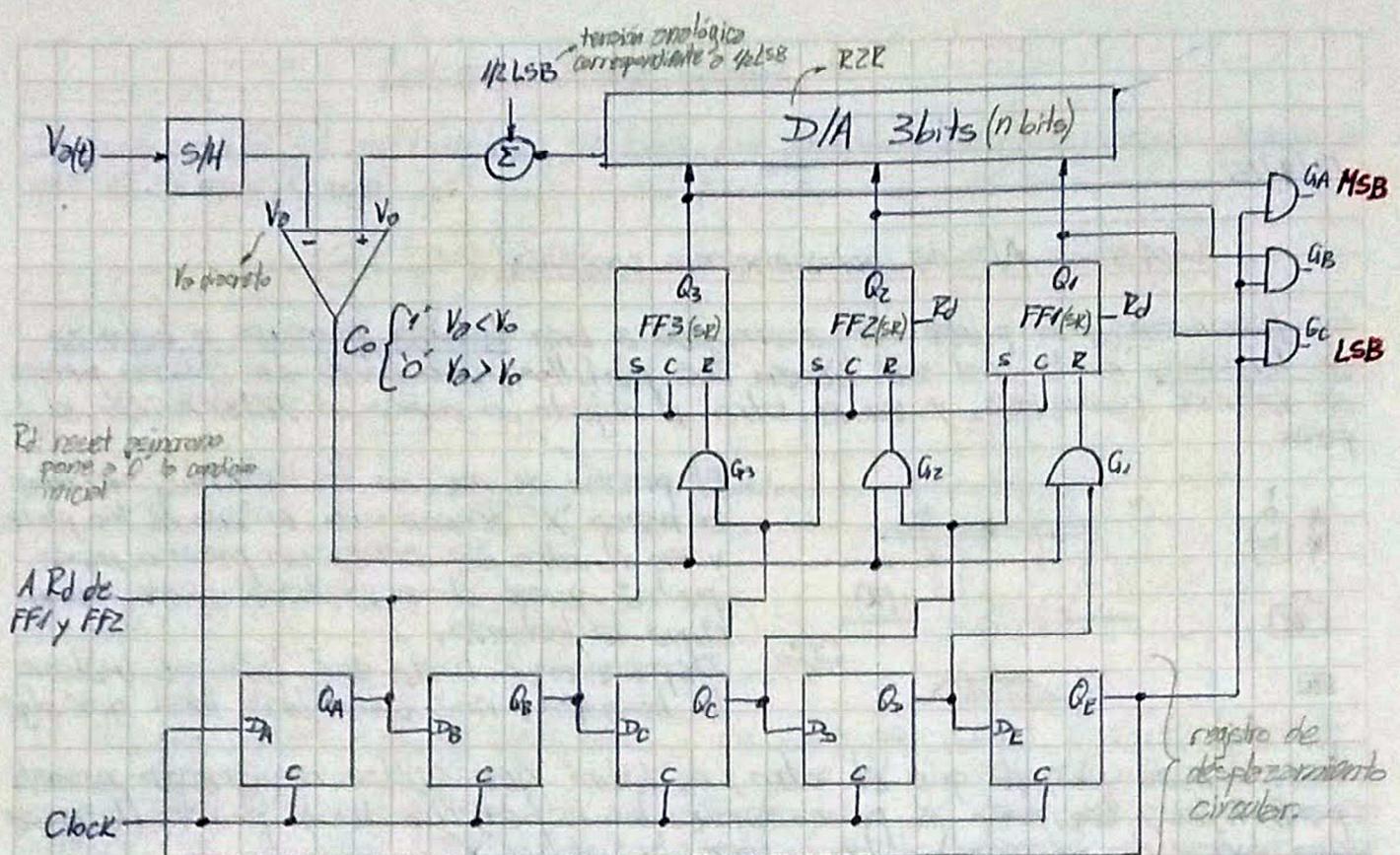
Los pesos son equivalentes a los bits, respecto a la analogía, la 1^{ra} medida, el MSB valdrá '1' y todos los demás '0', esto nos divide el problema en dos partes - "estoy en la parte alta o en la parte baja?"

Ejemplo de pesos:

min 00
1 0 0
1 1 0
1 0 1

Esquema circuital de un SAR.

In explicación del funcionamiento del circuito



S	R	Q_{in}	D	Q_{out}
0	0	Q_c	0	0
0	1	0	1	1
1	0	1		
1	1	X		

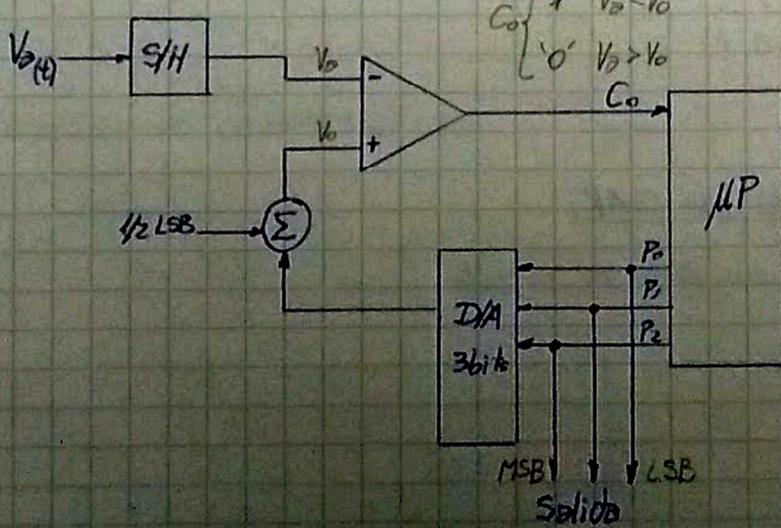
Cuando $Q_E = 1$
se muestra el resultado
en los salidas.

→ no significa que no se sabe; significa que el fabricante no garantiza un estado ('0' o '1') definido como en los anteriores casos.

El tiempo de conversión de este tipo de convertidor es de

$$T_C = n + 2$$

Este convertidor se presta para ser implementado con un μP ; algunas cosas de hardware serán útiles y otras serán implementadas por software. Por ejemplo:



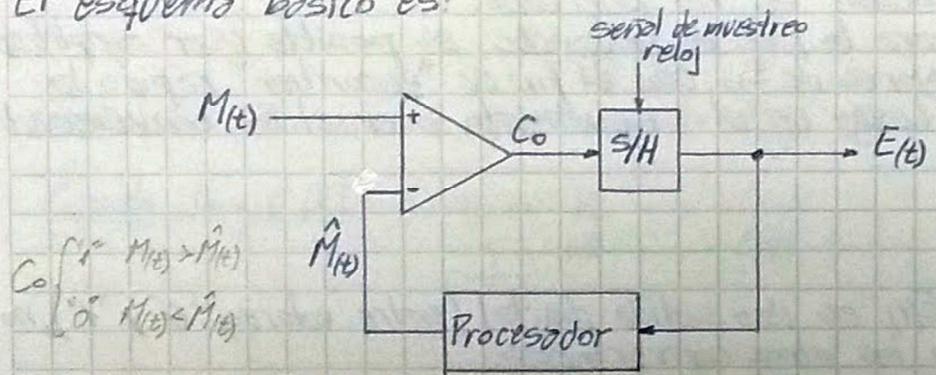
El diagrama de flujo siguiente corresponde a los pasos que debe seguir el µP para obtener la conversión A/D de la señal $V(t)$, por aproximaciones sucesivas en C y en Assembly.

22/9/15

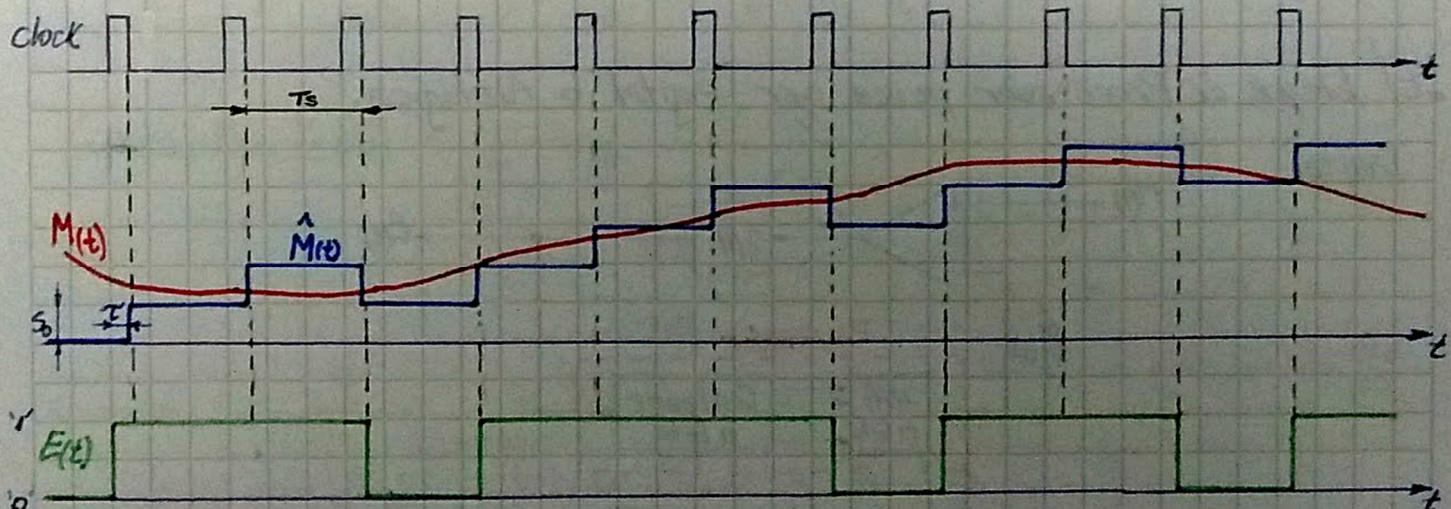
Conversor A/D de Seguimiento o Modulador Delta.

Este es el único cuya salida es digital pero no es binario natural, como lo eran los anteriores convertidores.

El esquema básico es:



Senales generadas:



Dónde:

$T_s = T_c$ - tiempo de clock o sampling

S_a - amplitud de cada nivel discreto o "escalón"

τ - tiempo de respuesta luego del frente descendente del clock

El comparador analógico, formado por el AO, constantemente está comparando $M_{(t)}$ contra $M_{(t)}$ (lo serial continuo, contra lo serial discreto). Si $M_{(t)} > M_{(t)}$, lo salido del AO es '1' y ante el pulso de clock, éste pasa de la entrada a la salida del SH. Así el Procesador incrementa su salido (valor de amplitud).

Esto se conoce como **fase de encapiche**. donde lo serial $M_{(t)}$ (conforme transcurren los pulsos de clock) se irá aproximando a lo serial $M_{(t)}$ hasta superarlo, lo cual produce en el comparador la salida '0'. Lo cual producirá que $M_{(t)}$ se decremente (la amplitud) y comience (por cada pulso de clock) el **seguimiento** de $M_{(t)}$ respecto de $M_{(t)}$.

Problemas:

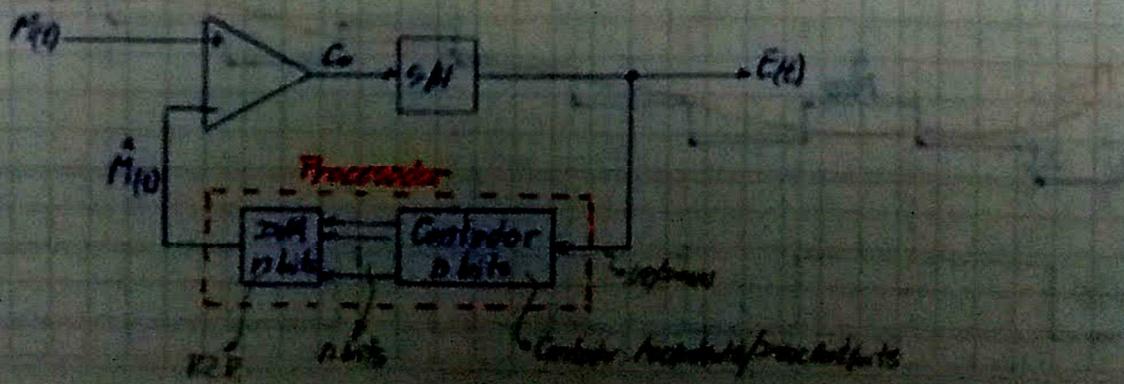
- Durante se produce el encapiche, por ej: una $M_{(t)}$ de variación rápida.
 - Solución: la frecuencia máxima en $M_{(t)}$ no puede ser mayor que la frecuencia del clock.
- Durante del escalón S_a no adecuado.
 - Solución: para la fase de encapiche, es posible usar amplitudes mayores de S_a con el fin de "encontrar" rápido lo serial $M_{(t)}$. Luego en el seguimiento usar el S_a convencional.

• La salida del conversor $E(t)$ es un salido digital (entre valores '0' y '1'), no siendo binaria natural como los otros conversores.

• Para volver a recuperar de $E(t)$ lo serial $M_{(t)}$ de manera sencilla es usar un circuito RC o un integrador con AO.

• El bloque de Procesador puede ser Digital o Analógico.

Digital



La cantidad de bits necesarios para el contador se calcula como:

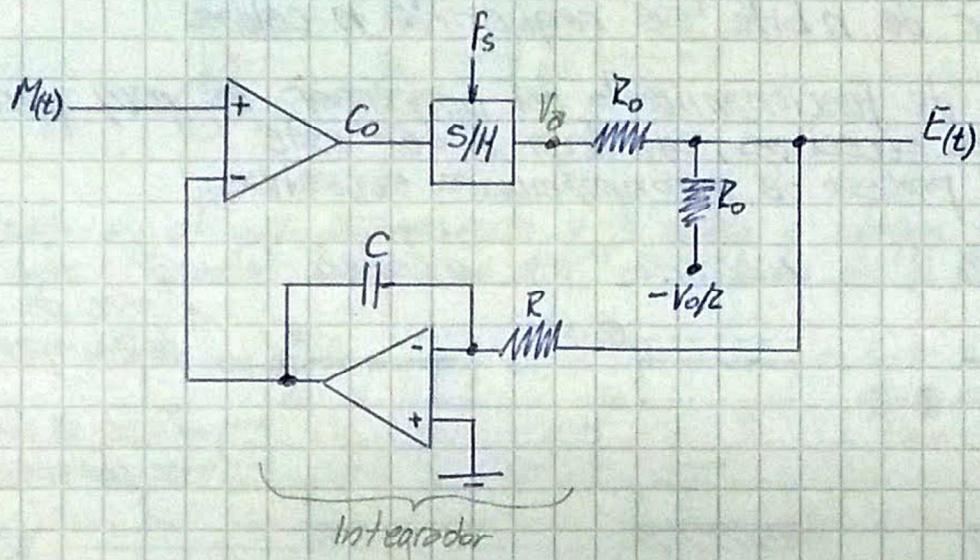
$$n = \frac{\ln\left(\frac{\text{Rango}}{S_0}\right)}{\ln(2)} \quad \rightarrow \text{Rango de la señal } M(t)$$

$$Z^n = \frac{\text{Rango}}{S_0}$$

El error del conversor lo determina la amplitud S_0 (tamaño del��bit) de la forma:

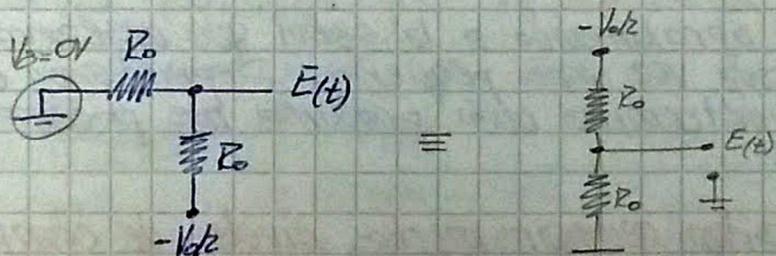
$$\pm \frac{S_0}{2} \quad \text{y error m醩imo} \Rightarrow S_0$$

An醠ogico:



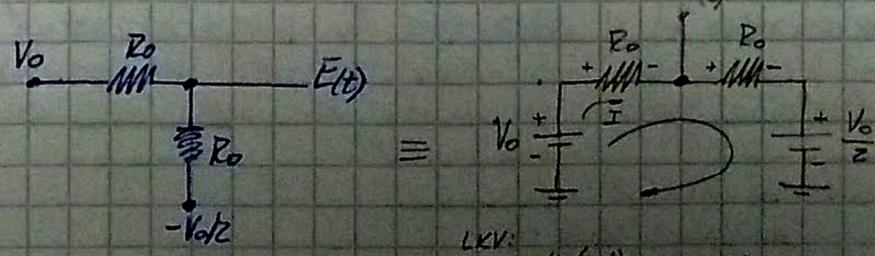
Para saber los valores que adopta $E(t)$, analizamos la rama de los R_o .

Cuando $V_a = 0 [V]$

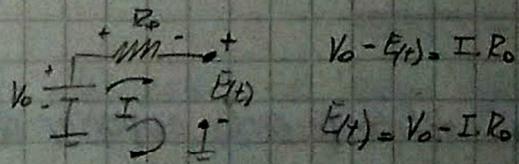


$$E(t) = -\frac{V_o}{2} \cdot \frac{R_o}{R_o + R_o} = -\frac{V_o}{2} \cdot \frac{R_o}{2R_o} = -\frac{V_o}{4}$$

Cuando $V_a = V_o [V]$:



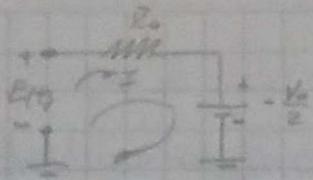
Si analizo



$$V_o - E(t) = I R_o$$

$$V_o - \left(-\frac{V_o}{2}\right) = I \cdot 2R_o \quad \therefore I = \frac{3V_o}{4R_o}$$

de igual forma:



$$E(t) - \left(\frac{V_0}{2}\right) = I R_2$$

$$E(t) + \frac{V_0}{2} = \frac{3V_0}{4} R_2$$

$$E(t) = \frac{3V_0}{4} - \frac{V_0}{2} = +\frac{V_0}{4}$$

• Comprobado
con MULTISIM

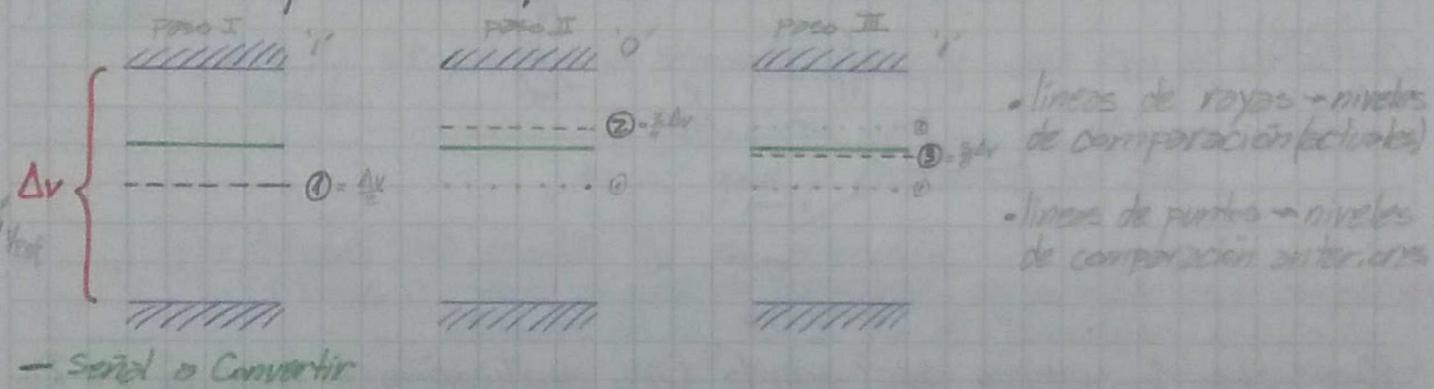
11/10/15

Conversor A/D Pipe-line. (modo corriente)

Este tipo de conversor veremos que está conformado por "cellos" o "bloques constructivos", cuyos sólidos representan 1 bit. Para obtener un conversor de n bits se requerirá n celdas.

El principio de funcionamiento del conversor es muy parecido al de aproximaciones sucesivas, sin llegar a ser éste.

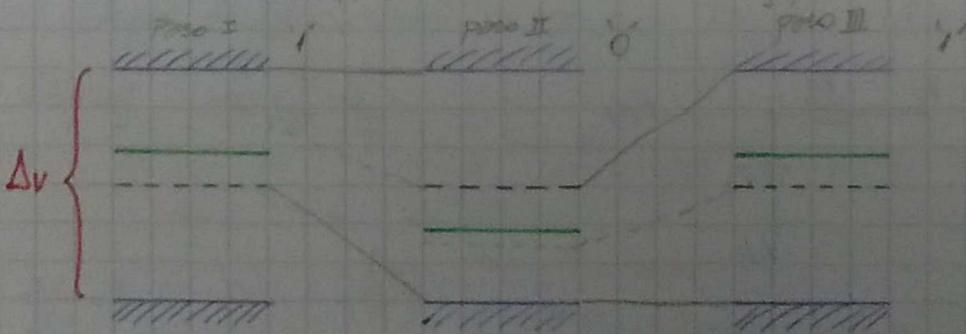
Recordando el proceso de aproximaciones sucesivas:



Nos vamos aproximando a la señal que deseamos convertir, haciendo los intervalos cada vez más pequeños. Dependiendo de la cantidad de bits, es b que nos determina cuán próximo nos podemos acercar a la señal a convertir.

"Se necesitan generar muchos niveles de comparación" \Rightarrow plantea dudas

* La base o idea del Pipe-line es evitar la cantidad de comparaciones. Teniendo siempre el mismo nivel de comparación, y variando el nivel de la señal a convertir (ya en el anterior caso era fijo)



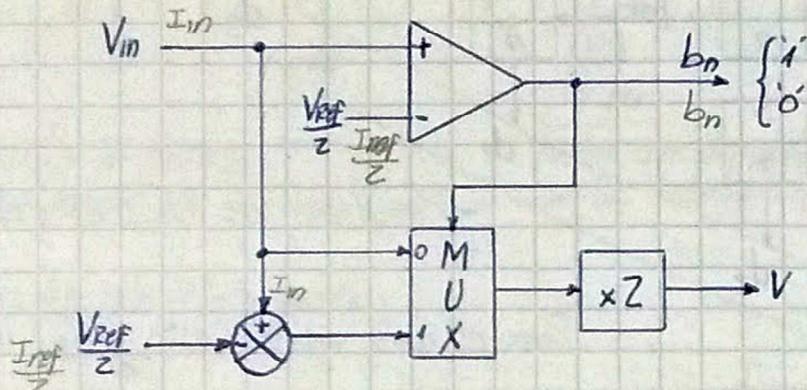
La situación del paso II es como un "zoom" del segmento superior del paso I.
 \rightarrow no es un zoom, es solo para interpretarlo.

Según sea el resultado de la comparación del paso I, a la señal se le sumará o restará un valor $\Delta V/2$, para luego hacer la comparación del paso II, y así sucesivamente. Otra cosa que se realiza entre pasos, es amplificar (o multiplicar x2) la señal o convertir para mantener la escala.

Diagrama en bloques:

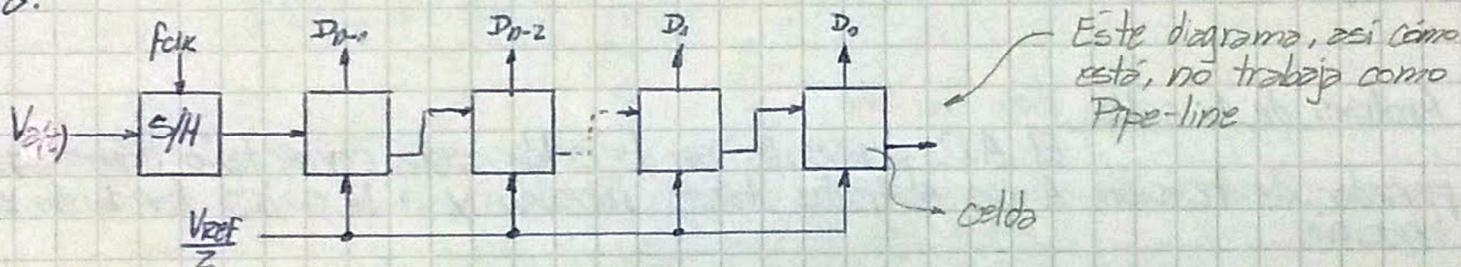
(en lápiz \Rightarrow modo corriente)

b_n : bit enésimo



Rango de corriente de entrada a convertir: $0 \sim I_{ref}$

Este diagrama es el que representa a la celda o bloque fundamental del ADC Pipe-line. Podemos hacer un ADC de n bits de la siguiente manera:



Para este diagrama, la fclk está dada por el tiempo de propagación de la celda. Entonces:

$$T_{clk(min)} = n \cdot T_{cell}$$

$$T_{clk(min)} \Leftrightarrow f_{clk(max)}$$

$\hookrightarrow n$ celdas = n bits

"Si bien funciona con un único pulso de clock para tener el resultado de la conversión, el tiempo empleado dependerá de la cantidad n de bits usados".

• Cómo se hace para que la conversión demore un único T_{cell} ?

→ Se coloca un S/H entre la salida de una celda y la entrada de otra, de ésta forma se obtiene el Pipe-line. (tendrá una latencia inicial hasta que la entrada llegue a la última celda, luego, por cada clk se obtiene un resultado).

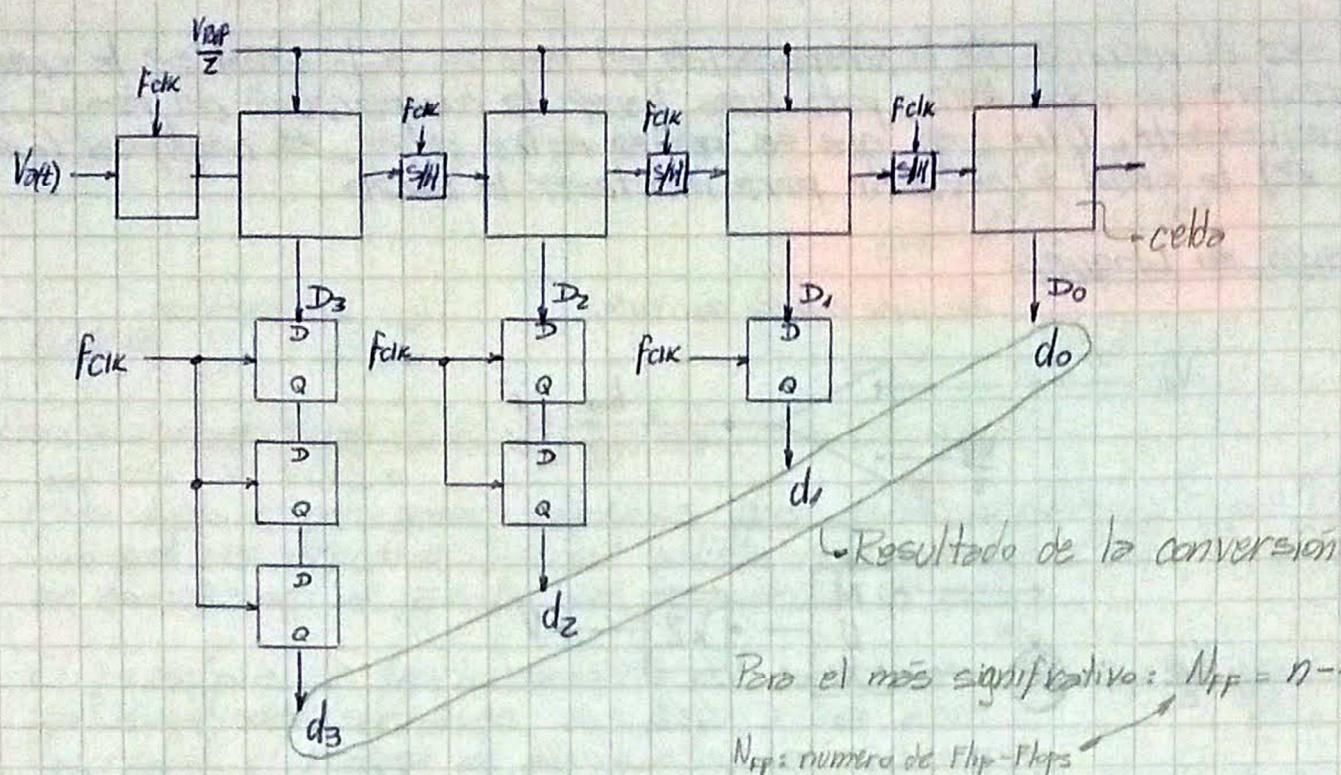
De esta forma:

"Todas las celdas, en todo momento se hallan trabajando"

$$T_c = T_{cell} = T_{clk}$$

Pero falta algo más, se deben colocar registros de desplazamiento para obtener de forma correcta los resultados de cada conversión. (conceptualmente se necesitan retardos)

Ejemplo de ADC Pipe-line de 4 bits



Para el más significativo: $N_{FP} = n - 1$

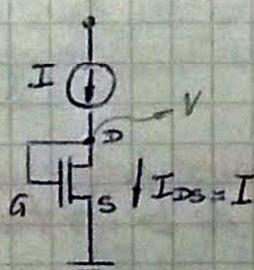
N_{FP} : número de Flip-Flops

"El error de la celda permite determinar el máximo número de bits permitidos?"

Analisis de la celda: modo corriente

El A.O. presente en la celda está conectado como comparador de tensión. A sus entradas tensión y a la salida también es tensión.

• Si quiero convertir corriente a tensión, se emplea un transistor MOSFET como diodo. ~ se puede entrar en zona lineal



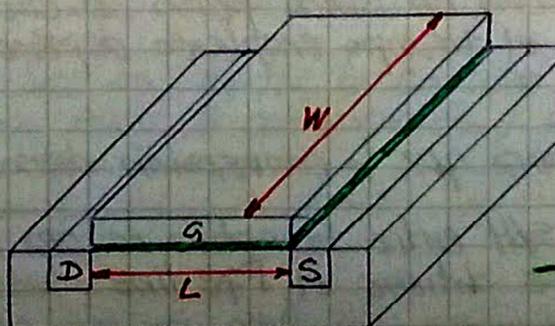
$$V = V_{GS}$$

$$I = I_{DDS}$$

$$I_{DDS} = \mu_n C_{ox} \frac{W}{L} \frac{(V_{GS} - V_{Th})^2}{2}$$

• El transistor (está validado para diodo)
saturado

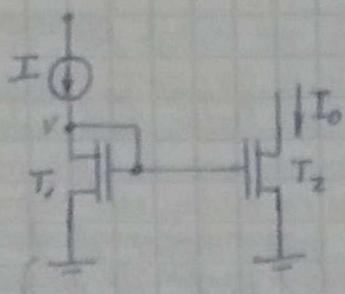
Mn: movilidad
Cox: capacidad del óxido
W: ancho
L: largo



Despejando V de la I_{DDS}

$$V = \sqrt{\frac{2 \cdot I}{\mu_n C_{ox} \frac{W}{L}}} + V_{Th}$$

Para obtener corriente desde una tensión, se emplea un espejo de corriente



conversión de corriente
a tensión

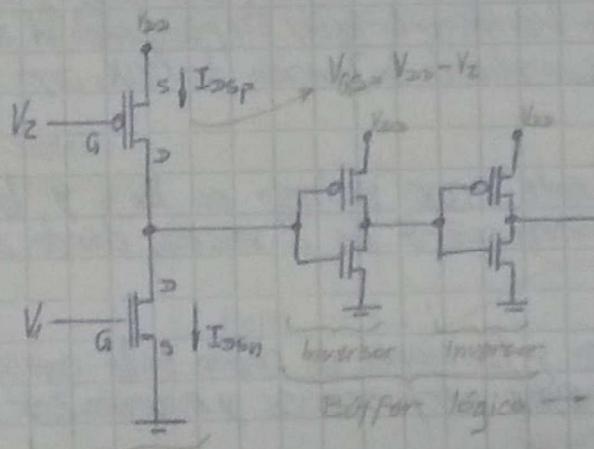
$$I_{O2} = \mu_n C_{ox} \left(\frac{W}{L} \right)_{T_2} \left[\frac{V_{GS2}}{\mu_n C_{ox} \left(\frac{W}{L} \right)_{T_1}} + V_{th} - V_{DS2} \right]$$

$$I_{O2} = \frac{\left(\frac{W}{L} \right)_{T_2}}{\left(\frac{W}{L} \right)_{T_1}} \cdot I = \text{en el bloque } [3]$$

factor que depende de la tasa
de los transistores

11/10/15

Comparador de Corriente:

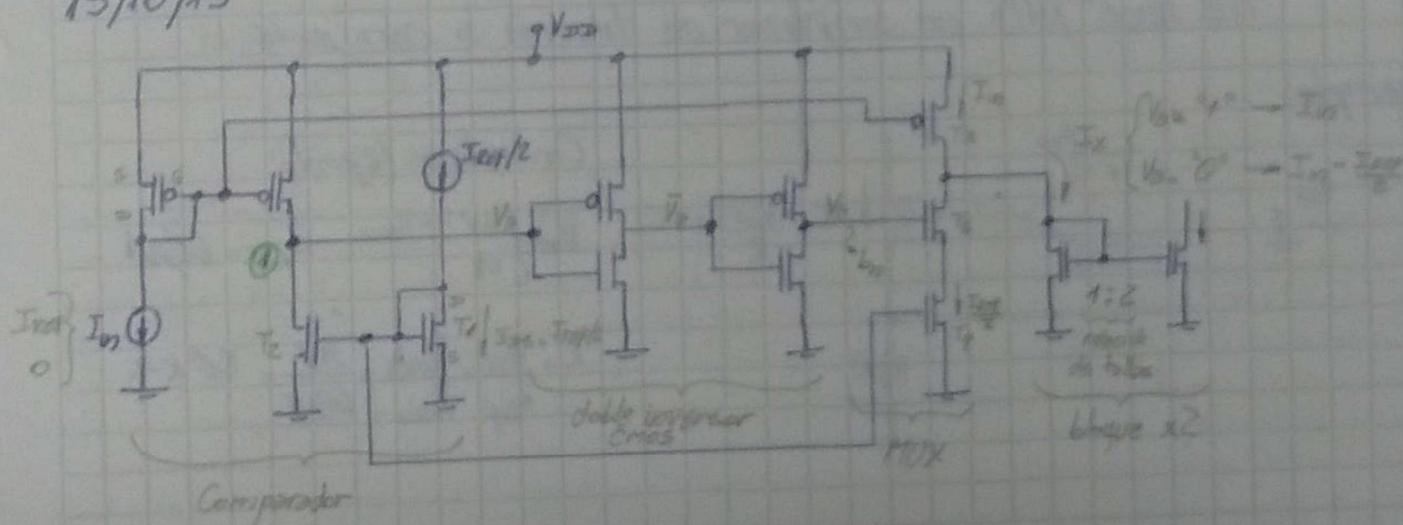


$$I_{O2p} = \mu_p C_{ox} \left(\frac{W}{L} \right)_{T_1} \frac{(V_{DS2} - V_{th})^2}{2}$$

Buffer lógico -- sirve para tener los niveles de tensión correctos
a lo lejos visto

en este comparador, "ganó" el de menor corriente

15/10/15



$$I_{DS2, T_2} = \mu_n C_{ox} \left(\frac{W}{L} \right)_{T_2} \frac{(V_{DS2} - V_{th})^2}{2} \quad \therefore V_{DS2} = \sqrt{\frac{2 I_{DS2}}{\mu_n C_{ox} \left(\frac{W}{L} \right)_{T_2}}} + V_{th} = \sqrt{\mu_n C_{ox} \frac{I_{ext}}{\frac{W}{L} T_2}}$$

min 28

• ¿Cuál es la corriente que circula por T_2 ?

$$I_{DSS T_2} = M_n \text{Cox} \left(\frac{W}{L} \right)_{T_2} \frac{(V_{GS} - V_{Th})^2}{Z}$$

dónde: $V_{GS} = V_{GSS T_1}$ → ecuación hallada anteriormente

Llegamos a:

$$I_{DSS T_2} = \frac{(W/L)_{T_2}}{(W/L)_{T_1}} - \frac{I_{Ref}}{Z}$$

→ la corriente que circula por T_2 , es una réplica o espejo de la corriente que circula por T_1 afectada por un factor debido a la tasa de los transistores.

min 31

El punto ① del circuito, en tensión, sube o baja para sacar de la saturación al transistor que quiere injectar o absorber la corriente de mayor magnitud.

Si bien, en este punto, se tienen niveles de tensión, no son niveles "lógicos"; para ello, se hace la etapa inversora CMOS, cuya salida tiene niveles de tensión aptos para las computadoras o sistemas lógicos.

min 30

El bloque MUX, consta de 3 transistores; T_3 y T_4 lo que hacen es copiar corriente y según sea el "factor de tasa" podrá aumentarla, reducirla o mantenerla igual. Dependiendo el estado de T_5 , éste dejará pasar el valor I_{in} o $I_{in} - \frac{I_{ref}}{2}$, cumpliendo la función de MUX, cuyo resultado es I_x .

min 48

El bloque xZ consiste en un "espejo de corriente" cuya entrada es I_x , y la relación de tasas de los transistores que lo componen será de 2 veces; dando a la salida un valor de corriente de $2 \cdot I_x$

min 50

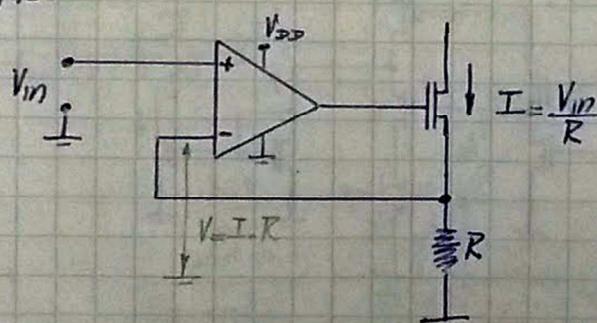
Lo visto hasta aquí es una versión mínima y funcional de cómo está constituido la celda, por lo que es posible hacerla de mayor performance y minimizar errores debido al copiado de corriente. (ejemplo fotocopia)

min 59

• ¿Qué sucede si el sensor puesto a la entrada, no entrega corriente pero sí tensión?

Me hace falta un conversor de tensión a corriente

Por ejemplo:



Con valores (Ejemplo):

$$V_{in} \begin{cases} 3V & -V_{max} \\ 0V & \end{cases} \quad I_{Ref} = 60\mu A \quad I_{max}$$

$$R = \frac{V_{in}}{I_{Ref}} = \frac{V_{max}}{I_{max}} = \frac{3V}{60\mu A} = 50k\Omega$$

16:24 min:

• ¿Cómo se hace el S/H con transistores?

Existe un problema al trabajar en modo corriente, ya que si $I_{in} = 0A$, todos los transistores deberían estar cortados. Para solventarlo, se aplica una corriente mínima de polarización, I_{B0} , donde se mantiene el nivel o rango de entrada I_{in} , a los fines que el circuito opere, incluso cuando $I_{in} = 0A$.

NOTA

1.1 Introducción

Mientras se incrementa el uso de técnicas de procesamiento de señales digitales en varias áreas de aplicación, mas se requiere el uso de conversores A/D CMOS que ocupen área de chip mínima y tasa de conversión alta. Los tres métodos de conversión mas usados en tecnología de procesamiento CMOS son el flash, el de aproximaciones sucesivas y el cíclico. La alternativa que aquí se presenta es un conversor que opera en forma similar al cíclico pero en arquitectura pipeline, y al cual se le suma la característica de operar en modo corriente en vez de tensión. Esto logra mejorar la precisión de la conversión ya que los errores de inyección de carga inducidos por las llaves no tienen efecto sobre la señal en sistemas que trabajan por corriente.

1.2 Conversor A/D pipeline en modo corriente

En la Fig. 1 y en la Fig. 2 se muestra un diagrama en bloque conceptual.

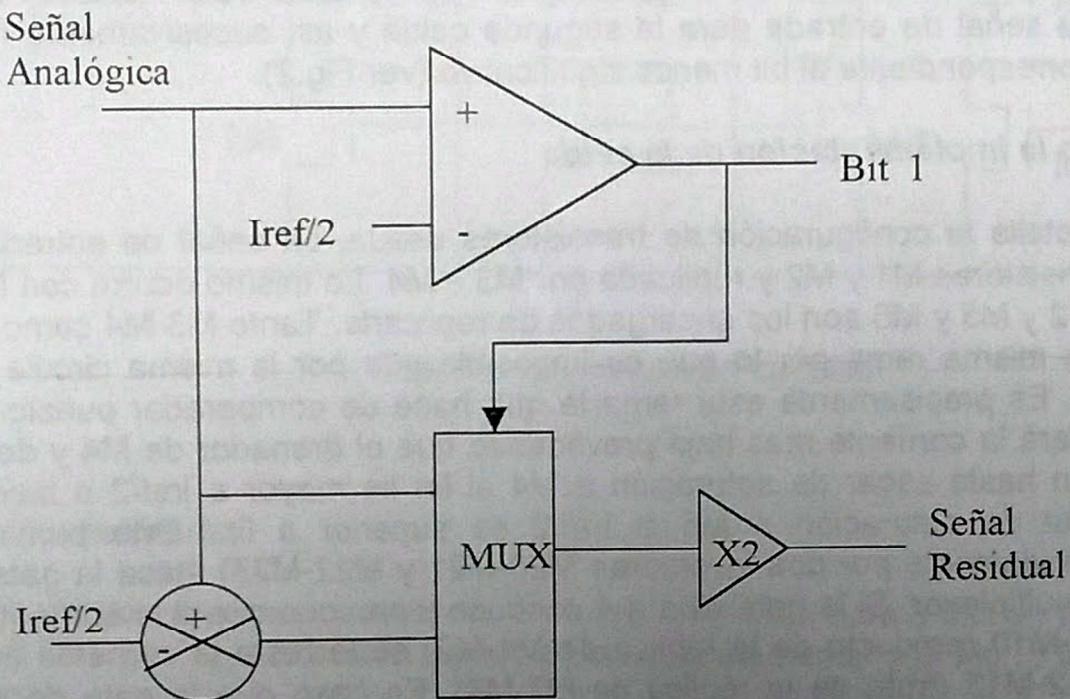


Fig. 1: Esquema de la celda de 1bit

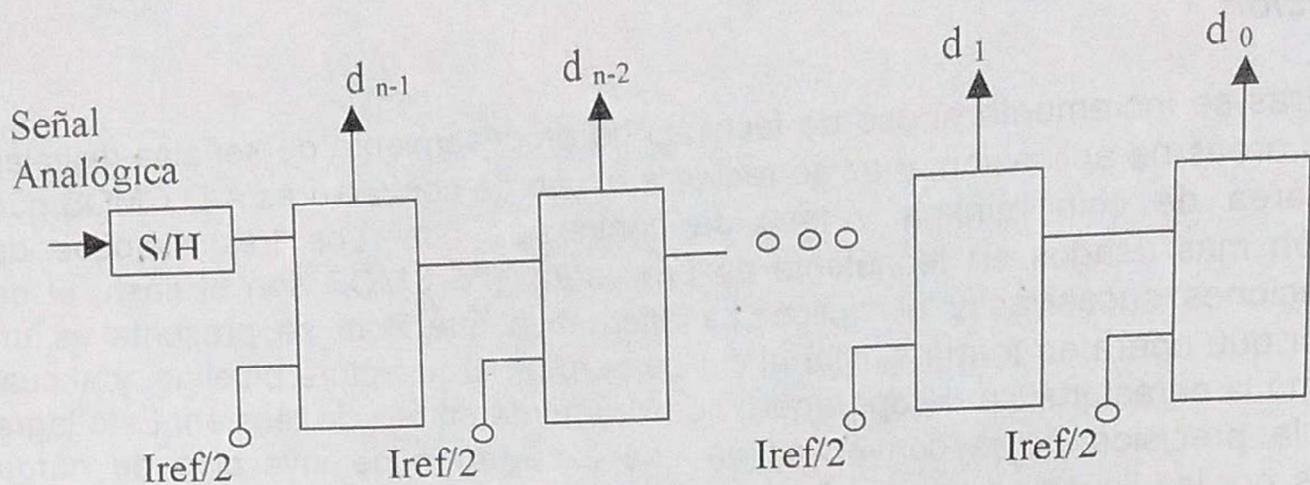


Fig. 2: Esquema del conversor total usando la celda de 1 bit en pipeline

La señal analógica tiene un rango entre 0 e I_{ref} . Se compara dicha señal con la $I_{ref}/2$ para determinar si se encuentra en la mitad inferior o superior del rango. Si se halla en la parte superior el bit de salida tendrá un valor de 1 y habilitará en el multiplexor el paso de la señal de entrada menos la $I_{ref}/2$, que luego es multiplicada por dos convirtiéndose en la señal residual de la celda. (ver Fig. 1). Este valor residual de la primera celda es la señal de entrada para la segunda celda y así sucesivamente hasta alcanzar la celda correspondiente al bit menos significativo (ver Fig.2).

1.3 Descripción de la implementación de la celda

La Fig. 3 detalla la configuración de transistores usada. La señal de entrada es copiada por los transistores M1 y M2 y replicada por M3 y M4. Lo mismo ocurre con M7 y M8 que copian $I_{ref}/2$ y M5 y M6 son los encargados de replicarla. Tanto M3-M4 como M5-M6 se hallan en la misma rama por lo que es imposible que por la misma circule dos corrientes distintas. Es precisamente esta rama la que hace de comparador puesto que por la misma circulará la corriente más baja provocando que el drenador de M4 y de M5 se eleve en tensión hasta sacar de saturación a M4 si I_{in} es mayor a $I_{ref}/2$ o baje su tensión hasta sacar de saturación a M5 si $I_{ref}/2$ es superior a I_{in} . Este punto de tensión(pasando previamente por dos inversores M20-M21 y M22-M23) ataca la gate de M11 que hace de multiplexor. Si la gate está a 1 conduce y provoca que a la corriente I_{in} que circula por M9-M10 (producto de la réplica de M1-M2) se le reste la corriente $I_{ref}/2$ que circula por M12-M13 (fruto de la réplica de M7-M8). En caso que la gate de M11 tenga un cero el transistor estará cortado y toda la corriente I_{in} circulará por M14-M15 que se hallan conectados como diodos. En ambos casos la corriente circulando por M14-M15 será replicada por un factor dos ya que M16-M17 y M18-M19 presentan la misma talla y están conectados en paralelo lo que asegura una mayor precisión en el copiado.

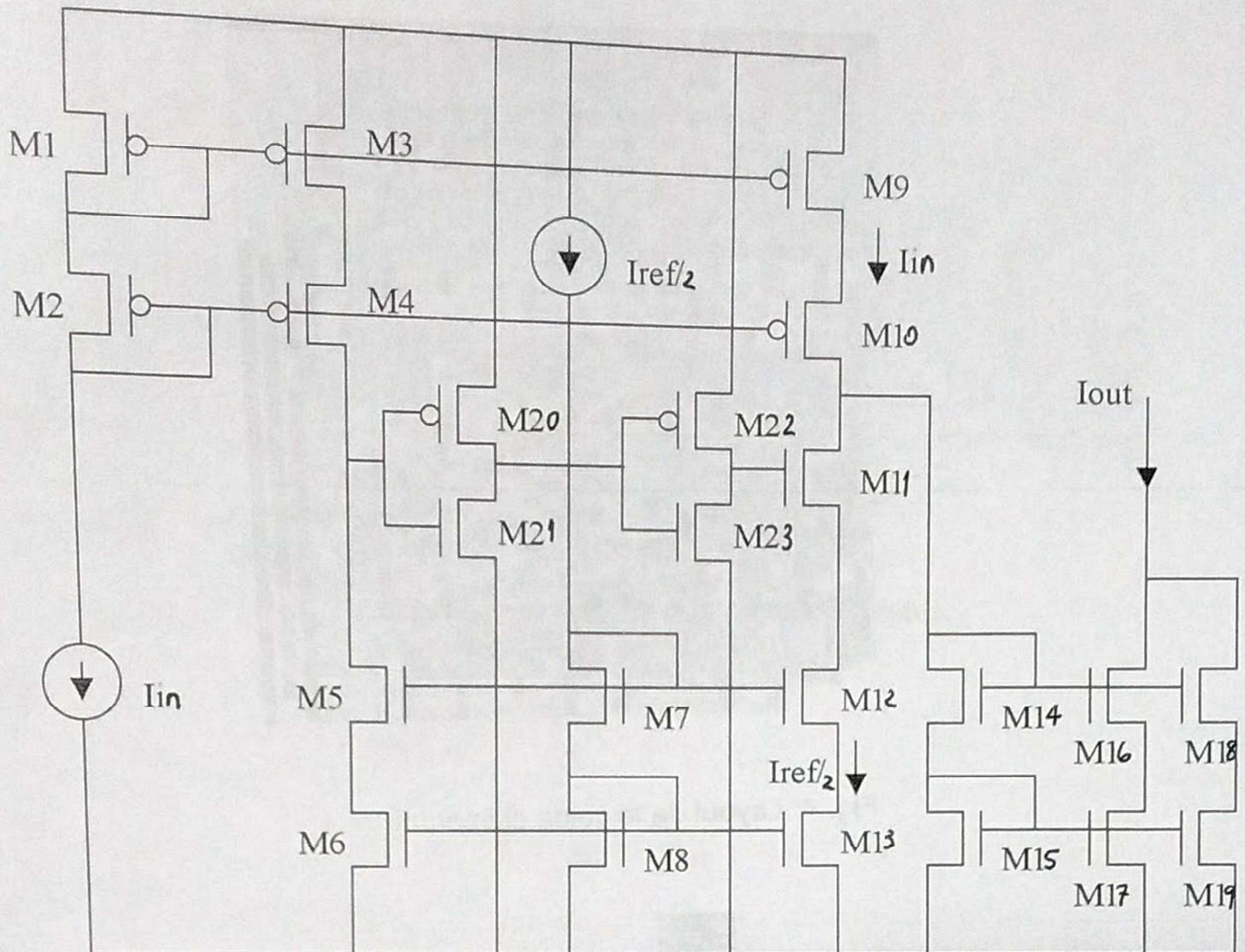


Fig. 3: Implementación circuitual

1.4 Implementación

Una versión de este conversor ha sido implementada para 8 bits en tecnología CMOS AMI 1.6 Um. La Fig. 4 ilustra el layout de la celda elemental y la Fig. 5 una vista total del conversor.

1.5 Conclusiones

Un A/D de 8 bits en modo corriente ha sido diseñado e implementado con la idea de incorporarlo a la plataforma básica para dispositivos inteligentes que se está gestando. Su arquitectura pipeline lo hace ideal para ser una celda de biblioteca y de acuerdo a la aplicación en la que se lo utilice se ajustará el número de bits del conversor. El testeo se hará después de su fabricación durante el año 2004.

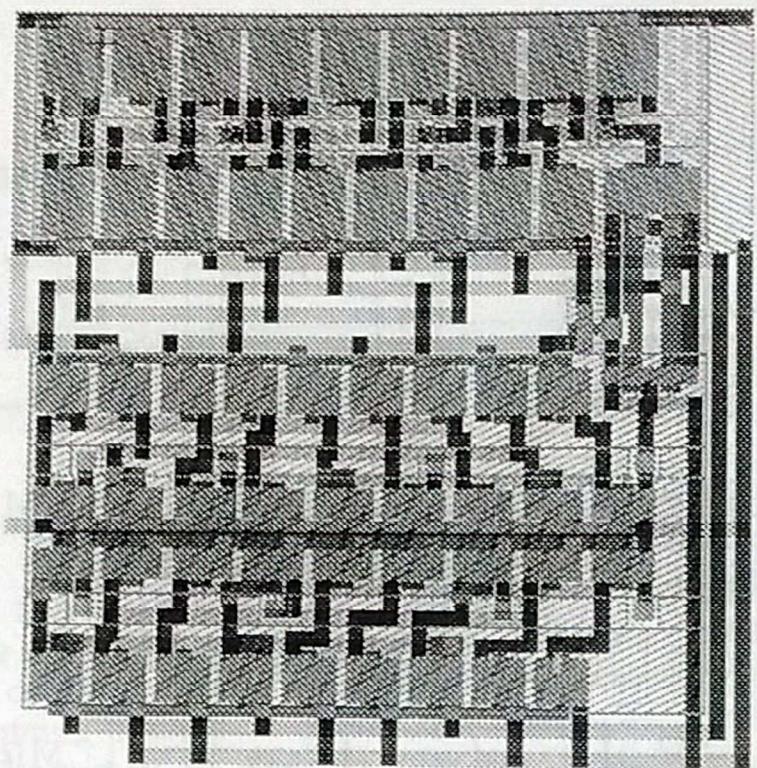


Fig. 4: Layout de la celda elemental

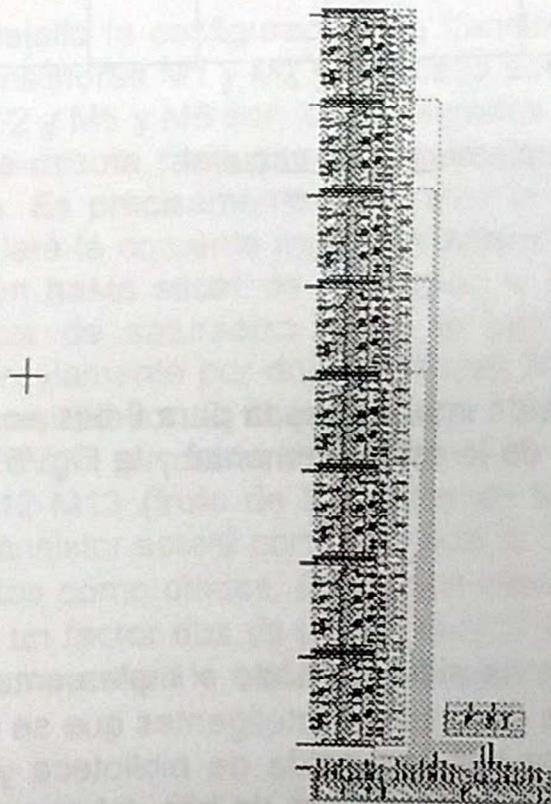
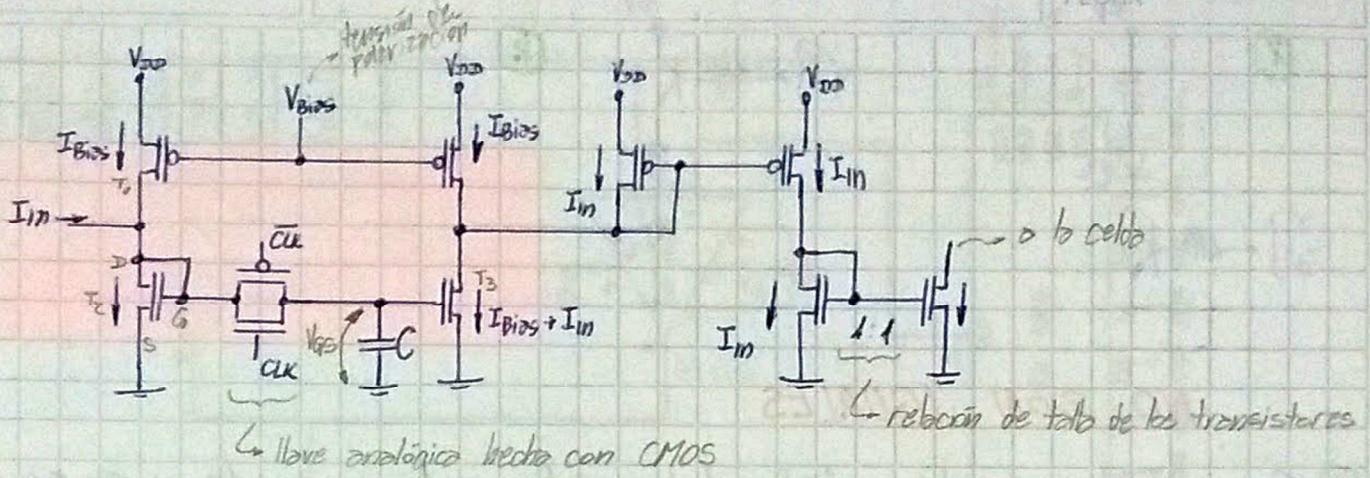


Fig. 5: Vista general del conversor A/D de 8 bits



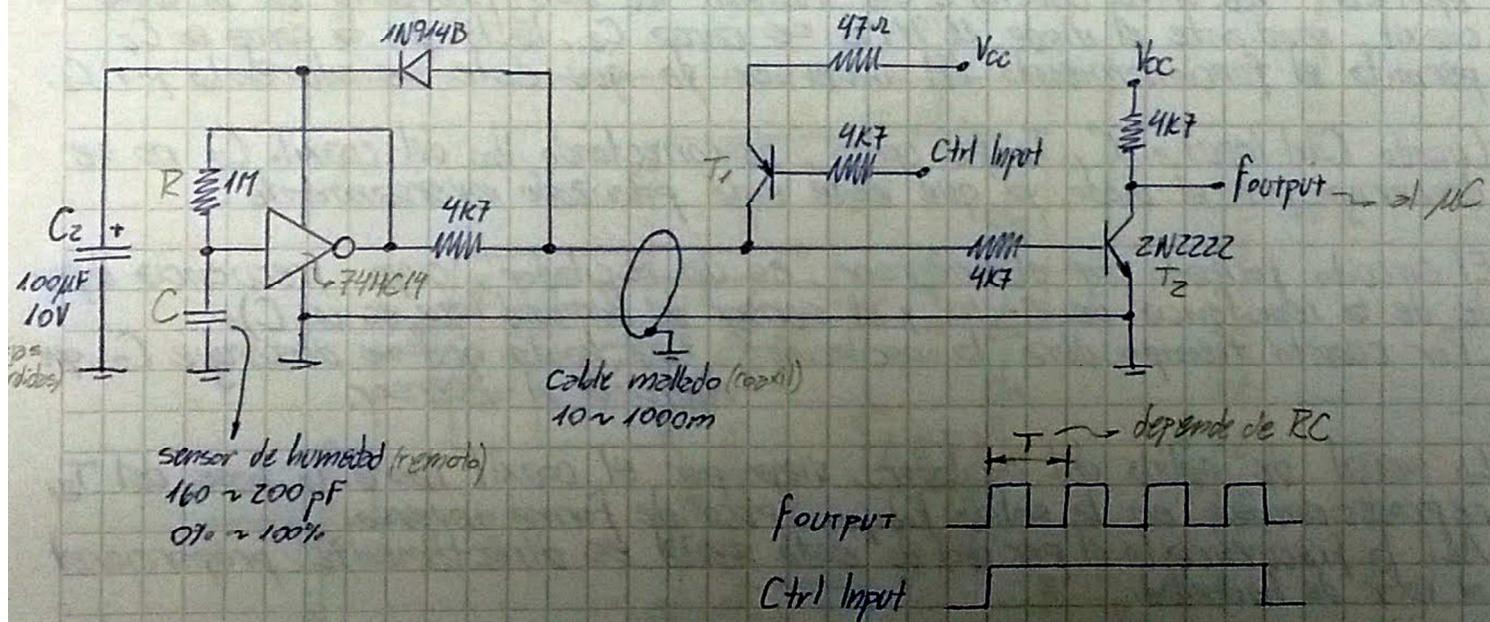
Cuando el $CLK = "1"$, se activan los 2 transistores de la llave, proporcionando un camino o la corriente hacia el capacitor C , cargándolo a $V_C = V_{BS}$.
Cuando el $CLK = "0"$, la llave se abre y C queda cargado.

Si T_2 y T_3 tienen la misma tasa, por T_3 circula $I_{Bias} + I_{in}$.

22/10/15

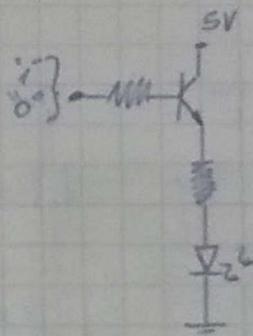
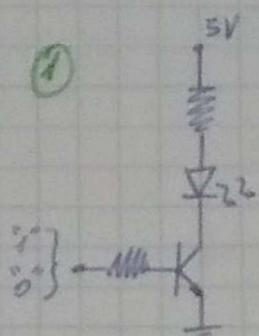
Conversor A/D de tensión a frecuencia (Anexo al temario del curso)

Este conversor no presenta una salida tipo binario natural, pero si será una señal digital cuya frecuencia depende de la señal de entrada, por ejemplo: temperatura, humedad u otra variable.



• El mismo circuito se puede emplear para medir temperatura en forma remota

min 20
Importante! ¿Los siguientes circuitos son iguales?



Se simula en MULTISIM
y ambos funcionan.
¿? probar en lo
realizado.

NO SON IGUALES

① Cuando la base está en "0", el transistor está en corte y en su colector están presentes los 5V de la fuente.
Cuando la base está en "1", el transistor se saturó y la masa "aparece" en el colector.

② Cuando la base está en "0", el transistor está en corte, no hay corriente por el led.
Cuando la base se pone en "1", el transistor se pone en saturación y conduce. Inmediatamente, los 5V de la fuente están presente en el emisor; la diferencia de potencial base-emisor es cero, llevando el transistor a corte.

Para solventar el problema de ②, el transistor debe ser un PNP.

Volviendo al conversor, cuando Ctrl Input = "0", T_1 se polariza, haciendo "aparecer" V_{oc} en el colector. Esta tensión V_{oc} está presente en el cable coaxil, mediante el diodo 1N914B se carga C_2 , hasta que la corriente en C_2 permite el funcionamiento del inversor ya que éste se alimenta por C_2 .

Cuando Ctrl Input = "1" T_1 se corta, desconectando V_{oc} del coaxil. C_2 no se descarga por el diodo, ya que éste está polarizado inversamente.

El circuito formado por el inversor, es un oscilador, cuya frecuencia dependerá de la resistencia de R_2 y el sensor de humedad (que es un C).
Por cuánto tiempo dura la oscilación? será hasta que se descargue C_2 , que alimenta al inversor.

La señal de salida del oscilador, viaja por el coaxil hacia la base del T_2 , reproduciéndose en la salida final, pero de forma inversa.
Así, la frecuencia (o el periodo) de este señal es directamente proporcional al valor de humedad.

Como se hace para medir temperaturas? se puede usar el mismo circuito, con la salvedad de que C sea fijo y R sea un termistor.

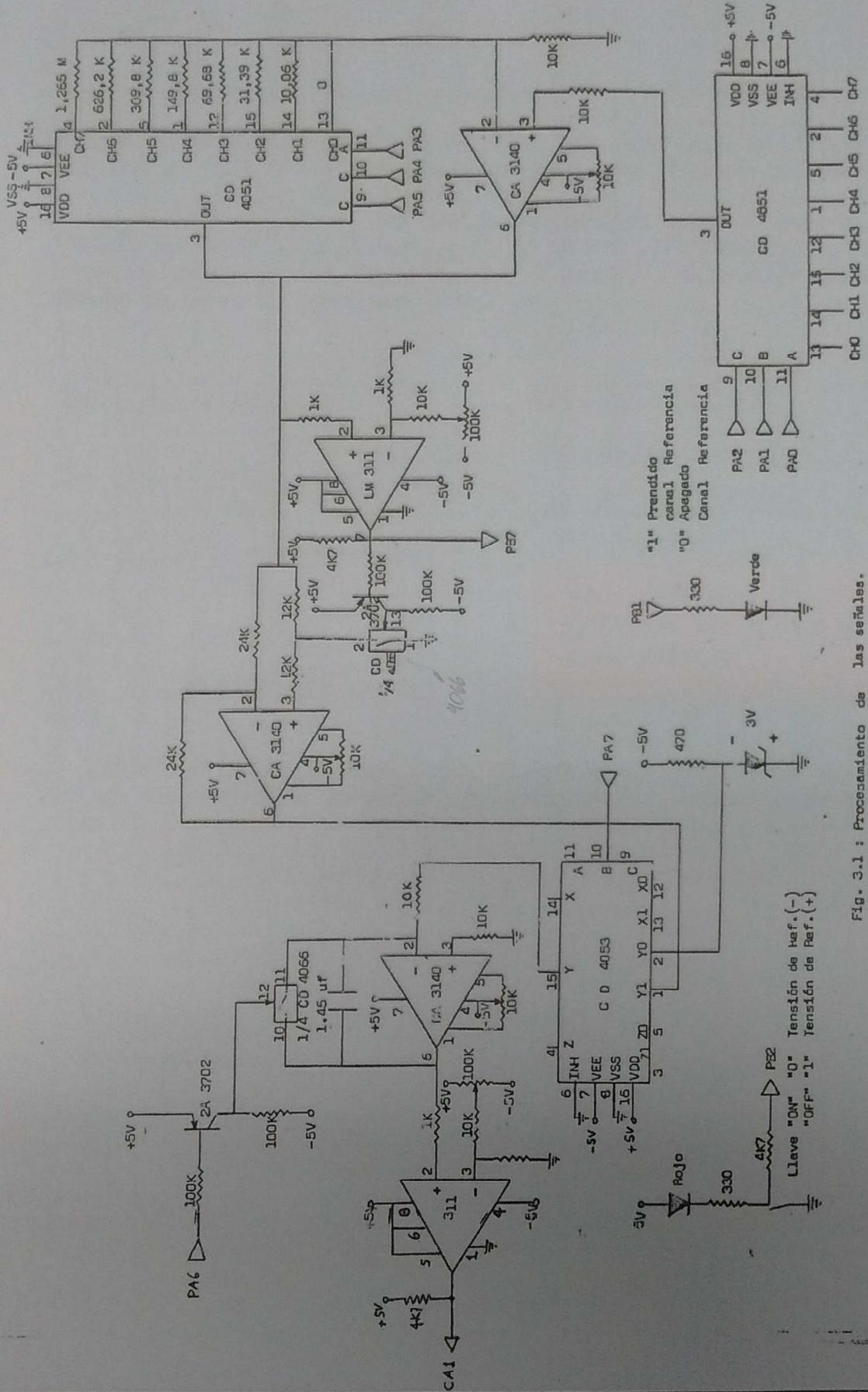


Fig. 3.1 : Procesamiento de las señales.

29/10/15

Sistemas de adquisición de datos (materia del curso)

Por ser un sistema de adquisición de datos, se tiene más de 1 canal; ello se hace con el MUX 4051; 8 canales combinados por 3 líneas de selección.

El conjunto 3140 y 4051 es un amplificador de ganancia variable. Así, es posible que cada señal (de cada canal) tenga una amplificación necesaria. Viendo los valores de los R que forman el circuito, es posible determinar los valores de ganancia para cada canal; así:

$$\frac{V_o}{V_{in}} = A_V = 1 + \frac{R_{in}}{R}$$



$$Ch 0: A_V = 1 + \frac{0.2}{10k} = 1$$

$$Ch 5: A_V = 31,98 \approx 32$$

$$Ch 6: A_V = 63,62 \approx 64$$

$$Ch 7: A_V = 127,5 \approx 128$$

$$Ch 2: A_V = 1 + \frac{31,39k}{10k} = 4,139 \approx 4$$

Para este caso, las ganancias son el doble del canal anterior.

$$Ch 3: A_V = 7,969 \approx 8$$

$$Ch 4: A_V = 15,98 \approx 16$$

mm m

El LM311 está funcionando como comparador; dependiendo si la señal de entrada es positiva o negativa, la salida está en el estado abierto, en 5v o 0v.

Nota porque ese dia tiene el parcial de resumen