S. R.

MINISTÉRIO DA DEFESA NACIONAL
FORÇA AÉREA PORTUGUESA
CENTRO DE FORMAÇÃO MILITAR E TÉCNICA

Curso de Formação de Praças - RC

COMPÊNDIO

ELECTRÓNICA DIGITAL 2

EPR: SAJ João Marques

CCF 335-17

Julho 2008

ENTIDADE FORMADORA ACREDITADA POR DGERT DIRECÇÃO-GERAL DO EMPREGO E DAS RELAÇÕES DE TRABALHO

S.  R.

MINISTÉRIO DA DEFESA NACIONAL
FORÇA AÉREA PORTUGUESA
CENTRO DE FORMAÇÃO MILITAR E TÉCNICA

CARTA DE PROMULGAÇÃO

JULHO 2008

1. O Compêndio de "Electrónica Digital 2" é uma Publicação "NÃO CLASSIFICADA".

2. Esta publicação entra em vigor logo que recebida.

3. É permitido copiar ou fazer extractos desta publicação sem autorização da entidade promulgadora.

O COMANDANTE

Vítor Manuel Alves Francisco

COR/PILAV

REGISTO DE ALTERAÇÕES

IDENTIFICAÇÃO DA ALTERAÇÃO, Nº DE REGISTO, DATA	DATA DE INTRODUÇÃO	DATA DE ENTRADA EM VIGOR	ASSINATURA, POSTO E UNIDADE DE QUEM INTRODUZIU A ALTERAÇÃO

Cursos:	Curso de Formação de Praças - RC
Nome do Compêndio:	Electrónica Digital 2
Disciplina:	Electrónica Digital e Técnicas Digitais
Data de elaboração:	Abril 2008
Elaborado Por:	SAJ/Meleca João Marques
Verificado Por:	Gabinete da Qualidade da Formação
Comando G. Formação:	TCOR / ENGAER José Saúde
Director de Área:	MAJ / TMMEL Abílio Carmo
Director de Curso:	TEN / TMMEL António Graveto
Formador:	SAJ/Meleca João Marques

ATENÇÃO:

Esta publicação destina-se a apoiar os formandos a frequentarem o Curso de Formação de Praças das especialidades MELECA, MELIAV e OPINF na disciplina de Electrónica Digital e Técnicas Digitais.

Não pretendendo ser uma publicação exaustiva do curso em questão, apresenta-se como uma ferramenta de consulta quer durante a duração do curso, quer após a sua conclusão.

ÍNDICE

CIRCUITO SEQUENCIAIS

O grande avanço da electrónica digital foi dado pelos circuitos sequenciais. Num circuito sequencial, o valor de uma saída depende não somente da combinação de valores das entradas, mas também do valor anterior, isto é, o valor que a saída tinha antes da aplicação da combinação de valores nas entradas.

O circuito elementar da lógica sequencial é conhecido pelo seu nome em inglês, Flip-Flop. Por definição, um Flip-Flop é um circuito que, contém:

Duas entradas R e S, ou J e K ou uma entrada D.

Pode ter uma entrada de controlo (clock), CK, ou Enable (En).

Tem duas saídas complementares, Q e $\overline{Q}$.

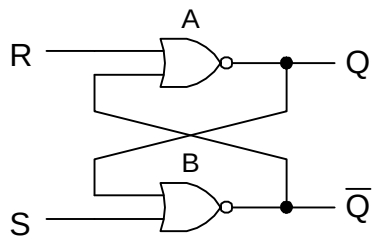
Pode ter uma entrada de Preset ou Set, (PR) e uma entrada de clear ou reset, (CL).

FLIP - FLOP RS

Pode-se construir facilmente um Flip-Flop com duas portas NAND ou NOR. O seu funcionamento está baseado nas suas tabelas de verdade.

O Flip – Flop com portas NOR.

A	B	S	Condição
0	0	1	--------
0	1	0	A saída é 0 quando pelo menos uma das entradas é 1
1	0	0	
1	1	0	



S	R	Q	$\overline{Q}$	Condição
1	0	1	0	Set
0	0	1	0	Mantém
0	1	0	1	Reset
0	0	0	1	Mantém
1	1	0	0	Inválido

Para efeitos de explicação do funcionamento do circuito as portas são designadas por A e B, as entradas são R e S, e as saídas são Q e $\overline{Q}$.

Para ver como o Flip-Flop tem dois estados, considere o que acontece se as entradas R=0 e S=1. Já vimos na tabela do NOR que uma entrada a 1 dá origem a 0 na saída. Neste caso a porta NOR B vai ter a saída $\overline{Q}$ =0.

A porta NOR A vai ter a entrada R=0 e a outra entrada também igual a 0, assim a saída Q=1.

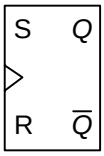
Quando S passa para 0 na porta B, a saída vai se manter, ou seja $\overline{Q}$ =0, porque entretanto a outra entrada ficou a 1.

Quando invertemos a condição e passamos a ter R=1, este fenómeno ocorre da mesma forma, mas os níveis de saída ficam invertidos.

Quando R=1 e S=1 em simultâneo, significa que estamos a activar a entrada SET e RESET em simultâneo e de acordo com a tabela de verdade do NOR, vamos ter ambas as saídas Q e $\overline{Q}$ igual a zero.

No Flip-Flop com portas NOR, as entradas estão normalmente a "0" e é o nível lógico 1 que faz activar o SET ou o RESET, ou seja o "1" na entrada S coloca a saída Q=1 e "1" na entrada R coloca a saída Q=0.

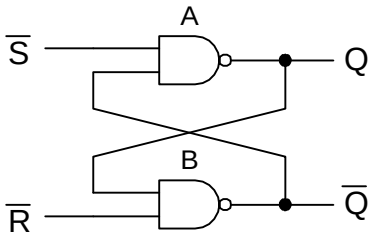
O símbolo do Flip-Flop RS é o seguinte:



O Flip – Flop com portas NAND.

A	B	S	Condição
0	0	1	A saída é 1 quando pelo menos uma das entradas é 0
0	1	1	
1	0	1	
1	1	0	-------

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	Condição
0	1	1	0	Set
1	1	1	0	Mantém
1	0	0	1	Reset
1	1	0	1	Mantém
0	0	1	1	Inválido



Para efeitos de explicação do funcionamento do circuito as portas são designadas por A e B, as entradas são $\overline{S}$ e $\overline{R}$, e as saídas são Q e $\overline{Q}$.

Para ver como o Flip-Flop tem dois estados, considere o que acontece se as entradas $\overline{R}$ =1 e $\overline{S}$ =0. Já vimos na tabela do NAND que uma entrada a 0 dá origem a 1 na saída. Neste caso a porta NAND A vai ter a saída Q=1.

A porta NAND B vai ter a entrada $\overline{R}$ =1 e a outra entrada também igual a 1, assim a saída $\overline{Q}$ =0.

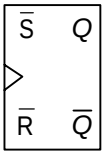
Quando $\overline{S}$ passa para 1 na porta A, a saída vai se manter, ou seja Q=1, porque entretanto a outra entrada ficou a 0.

Quando invertemos a condição e passamos a ter $\overline{R}$ =0, este fenómeno ocorre da mesma forma, mas os níveis de saída ficam invertidos.

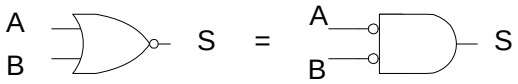
Quando $\overline{S}$ =0 e $\overline{R}$ =0 em simultâneo, significa que estamos a activar a entrada $\overline{SET}$ e $\overline{RESET}$ em simultâneo e de acordo com a tabela de verdade do NAND, vamos ter ambas as saídas Q e $\overline{Q}$ igual a um.

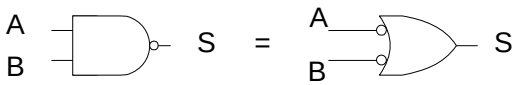
No Flip-Flop com portas NAND, as entradas estão normalmente a “1” e é o nível lógico 0 que faz activar o $\overline{SET}$ ou o $\overline{RESET}$, ou seja o “0” na entrada $\overline{S}$ coloca a saída Q=1 e “0” na entrada $\overline{R}$ coloca a saída Q=0.

O símbolo do Flip-Flop $\overline{S}\,\overline{R}$ é o seguinte:

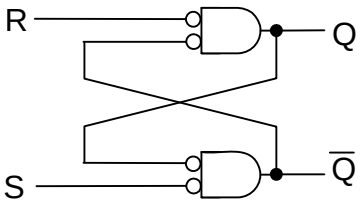


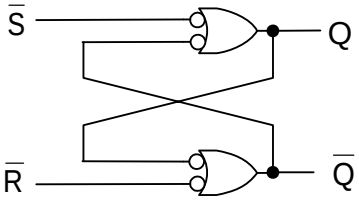
De acordo com as leis de De Morgan, podemos converter uma função NAND para OR e uma função NOR para AND.

$\overline{A+B} = \overline{A} \cdot \overline{B}$ 

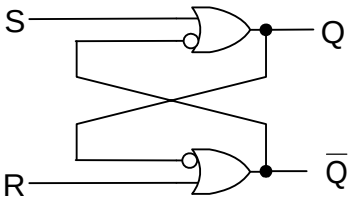
$\overline{A \cdot B} = \overline{A} + \overline{B}$ 

Desta forma podemos realizar dois circuitos Flip-Flop RS e $\overline{S}\,\overline{R}$, com o mesmo funcionamento dos

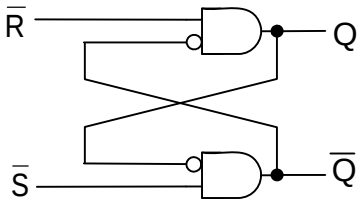




anteriores, mas usando portas lógicas AND e OR.



Podemos simplificar cada um dos circuitos, e retirar o inversor das entradas:



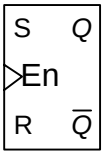
Estes dois circuitos têm novas tabelas de verdade e são as seguintes:

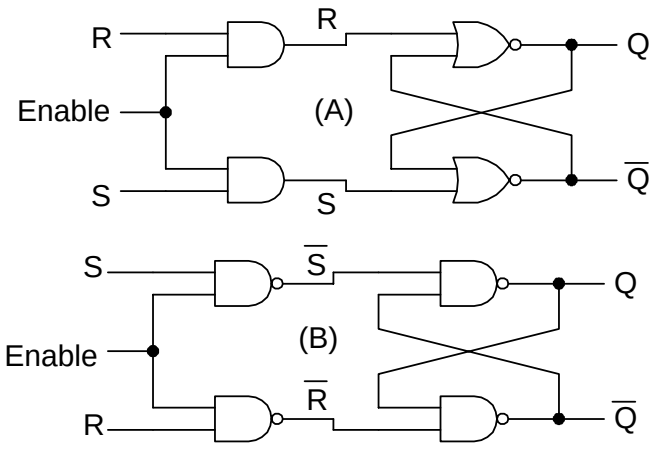
Flip-flop $\overline{S}\,\overline{R}$ (AND)					Flip-flop RS (OR)				
$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$	Condição	S	R	Q	$\overline{Q}$	Condição
0	1	1	0	Set	1	0	1	0	Set
1	1	1	0	Mantém	0	0	1	0	Mantém
1	0	0	1	Reset	0	1	0	1	Reset
1	1	0	1	Mantém	0	0	0	1	Mantém
0	0	0	0	Inválido	1	1	1	1	Inválido

Flip-flop RS com enable.

O flip-flop RS com enable pode ser usado para armazenar informação. Os dados colocados nas entradas RS só accionam o flip-flop quando a entrada enable for colocada a "1". Quando o Enable é igual a zero, O flip-flop RS está sempre na condição de manter o estado anterior.

Símbolo do Flip-Flop RS com Enable.

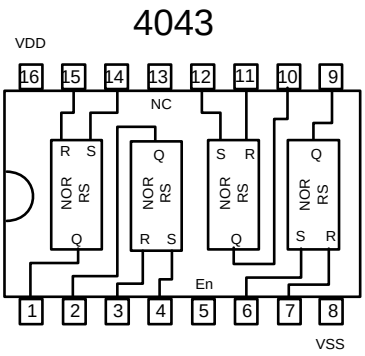




Entradas			Saídas (A)		Saídas (B)		Condição
Enable	S	R	Q	$\overline{Q}$	Q	$\overline{Q}$	
1	1	0	1	0	1	0	Set
1	0	0	1	0	1	0	Mantém
0	0	1	1	0	1	0	Mantém
1	0	1	0	1	0	1	Reset
1	0	0	0	1	0	1	Mantém
0	1	0	0	1	0	1	Mantém
1	1	0	1	0	1	0	Set
1	1	1	0	0	1	1	Inválido

O circuito A tem um comportamento semelhante ao circuito B, excepto na condição inválida. O valor da condição inválida (ilegal) está de acordo com as portas lógicas usadas no circuito, conforme a tabela de verdade seguinte:

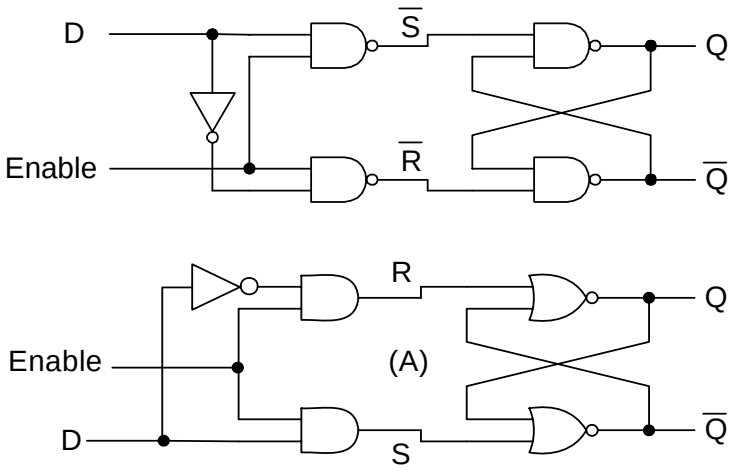
Este é um circuito integrado da família CMOS que contém 4 flip-flops RS com portas NOR. Existe uma entrada de enable no pino 5 que é comum a todos os flip-flops.



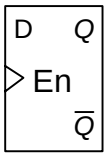
LATCH D

Se analisarmos a tabela de verdade do circuito anterior, podemos verificar que a saída muda apenas quando a entrada enable é igual a "1". Podemos também verificar que as entradas R e S devem estar sempre complementadas para se colocar as saídas no estado pretendido. Devido a isto podemos alterar os circuitos anteriores de modo a haver apenas uma entrada de dados, conforme os circuitos seguintes:

Neste caso temos dois Latch D, um construído com portas NAND e outro construído com portas NOR.

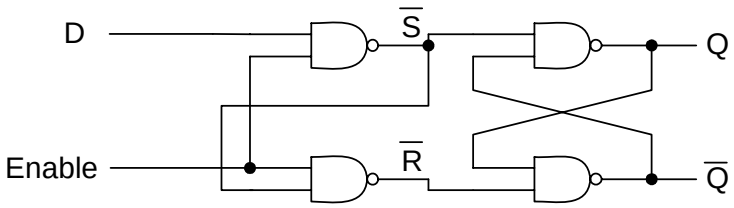


Ambos os circuitos têm a mesma tabela de verdade porque a condição inválida deixou de existir.

Símbolo	Entradas		Saídas		
	Enable	D	Q	$\overline{Q}$	Condição
	1	1	1	0	Set
	1	0	0	1	Reset
	1	1	1	0	Set
	0	1	1	0	Mantém
	0	0	1	0	Mantém
	0	1	1	0	Mantém
	1	0	0	1	Reset

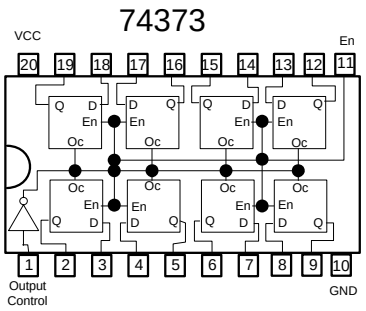
Então o Latch D é uma célula que permite ou não, a mudança da saída Q em função da entrada D e enable. Sempre que enable igual a zero, a saída Q Mantém sempre o estado anterior.

Existe uma outra versão para o Latch D:



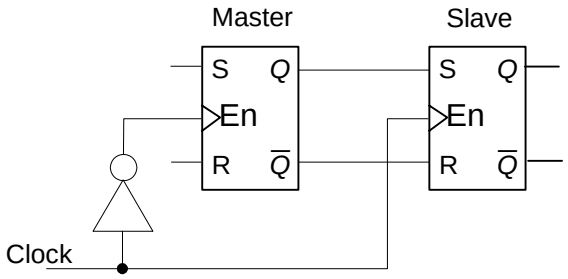
A substituição do inversor pela ligação ao ponto $\overline{S}$ é possível. Quando Enable é igual a zero, o ponto $\overline{S}$ e $\overline{R}$ são sempre igual a 1, desta forma o circuito mantém o estado anterior. Quando Enable igual a um. O ponto $\overline{S}$ é igual a $\overline{D}$, deste modo, este ponto de entrada tem o mesmo sinal que teria se estivesse ligado ao ponto D com um inversor.

Dentro dos vários modelos de circuitos integrados, existe um modelo da família lógica TTL com oito Latch's tipo D, conforme a seguinte figura:



RS MASTER SLAVE

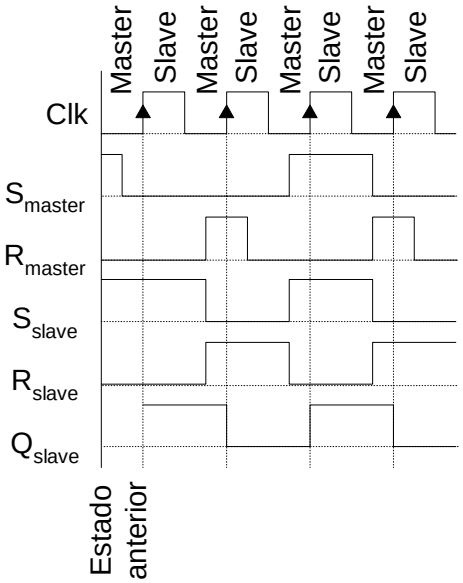
O flip-flop RS master slave, na realidade são dois flip-flop ligados em cascata, o slave está após o master.



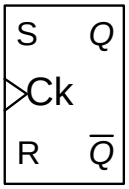
Quando o clock é igual a zero, está activado o enable do master, quando o clock é igual a um, está activado o enable do slave.

As saídas do master ligam nas entradas do slave, por isso a saída Q do master é a entrada S do slave.

Quando há mudança de estado do clock, transição ascendente, ou seja mudança de zero para um, a saída do slave assume o seu estado. O nível lógico da saída Q_{slave} depende do estado da entrada S_{master} antes de ter havido transição ascendente.



O símbolo do flip-flop RS master slave é:

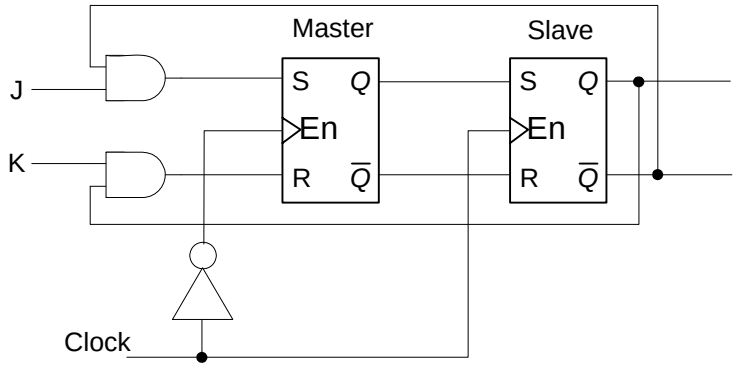


As entradas S_{master} igual a um e R_{master} igual a um, podem provocar uma condição inválida no flip-flop master quando enable igual a zero.

Tabela de verdade do flip-flop RS master slave					
S	R	Q	$\overline{Q}$	Clock	Condição
1	0	1	0	↑	Set
0	0	1	0	↑	Mantém
0	1	0	1	↑	Reset
0	0	0	1	↑	Mantém
1	1	x	x	↑	Inválido

JK MASTER SLAVE

O flip flop JK master slave, tem um comportamento semelhante ao flip flop SR master slave, J é o Set e K é o Reset, mas a condição J=K=1 assume a condição de inversão (Toggle).



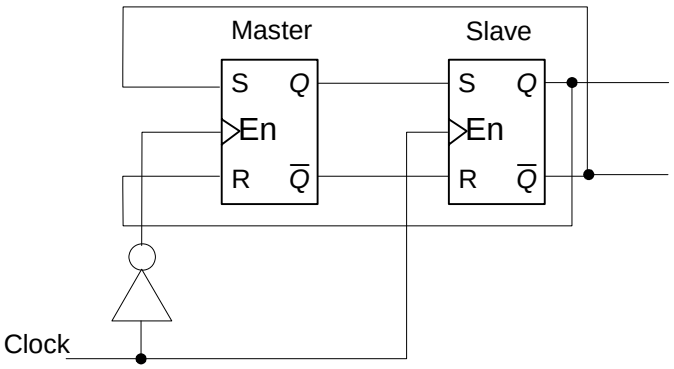
Quando J=1 e K=1, temos uma condição de Toggle. Esta condição provoca a inversão das saídas Q e $\overline{Q}$ em cada transição de clock.

Neste caso podemos simplificar as entradas J e K recorrendo a um dos postulados (a.1=a).

$S = J \cdot \overline{Q}$ $S = \overline{Q} \cdot 1 = \overline{Q}$

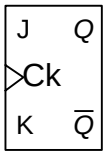
$R = K \cdot Q$ $R = Q \cdot 1 = Q$

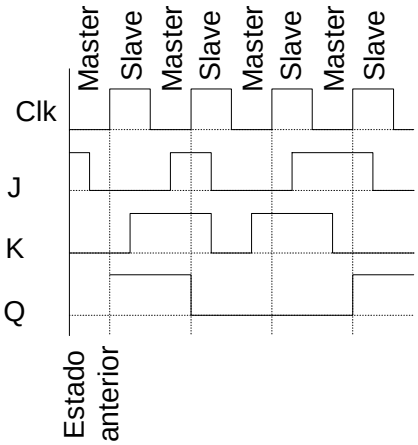
Nesta condição podemos simplificar o circuito:



e obtemos o seguinte diagrama temporal:

Símbolo:



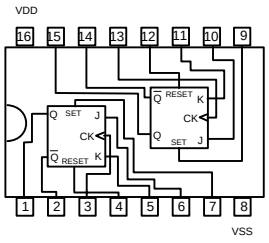


A tabela de verdade do flip-flop JK é a seguinte:

Tabela de verdade do flip-flop JK master slave					
J	K	Q	$\overline{Q}$	Clock	Condição
1	0	1	0	↑	Set
0	0	1	0	↑	Mantém
0	1	0	1	↑	Reset
0	0	0	1	↑	Mantém
1	1	1	0	↑	Inverte
1	1	0	1	↑	Inverte

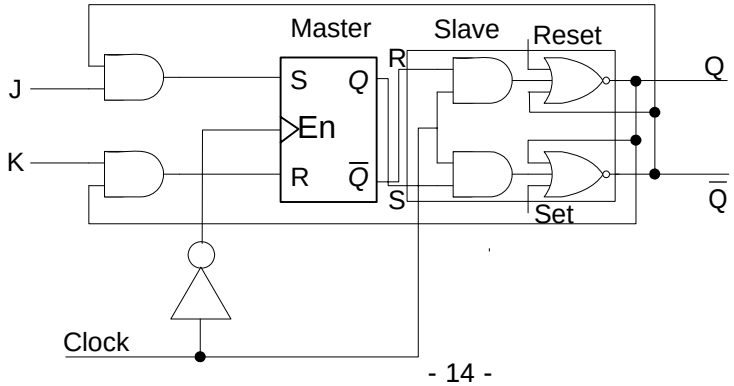
Um circuito integrado muito usado é o 4027 da família CMOS.

4027



Este componente tem dois flip-flop JK com duas entradas Set e Reset. Estas entradas não estão sincronizadas com a entrada de clock, desta forma o Set e o Reset podem alterar a saída Q em qualquer momento diferente do clock.

Para que as entradas funcionem num modo assíncrono, independentes do clock, elas têm que estar ligadas na saída do flip-flop, conforme o esquema:

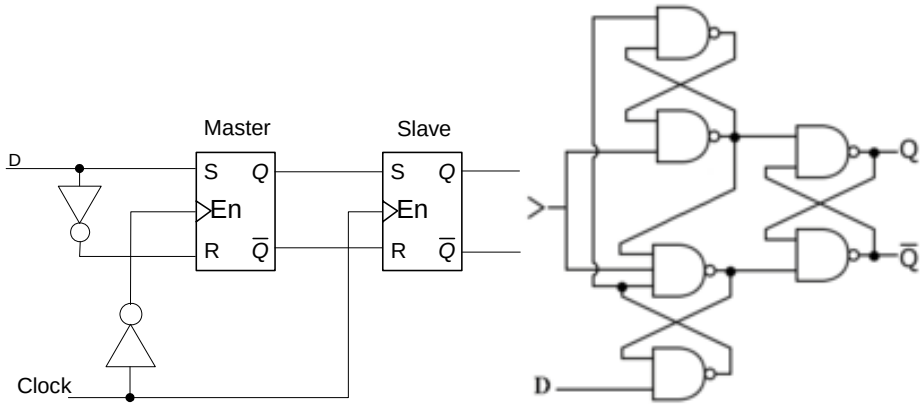


FLIP FLOP D

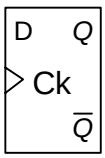
A saida Q do flip flop assume o valor da entrada D no instante da transição de clock. O flip-flop pode funcionar na transição ascendente (rising edge) ou na transição descendente (falling edge), mas nunca nas duas ao mesmo tempo.

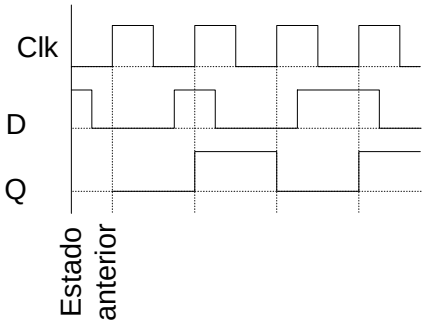
A vantagem deste flip-flop relativamente ao Latch D é que a entrada D só passa para a saída Q no instante da transição de clock.

O flip-flop D pode ser construído de duas maneiras diferentes. Podemos usar um flip-flop master slave e ligar um inversor nas entradas S e R ou usar o circuito do Edge-triggered flip-flop.



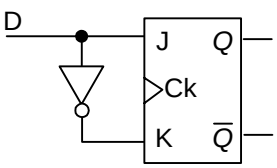
O símbolo do flip-flop D é:





O diagrama temporal é o seguinte:

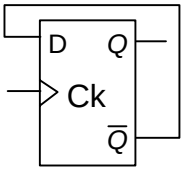
Podemos obter um flip-flop D a partir de um flip-flop JK master slave, ligando as entradas J (Set) e K (Reset) através de um inversor.

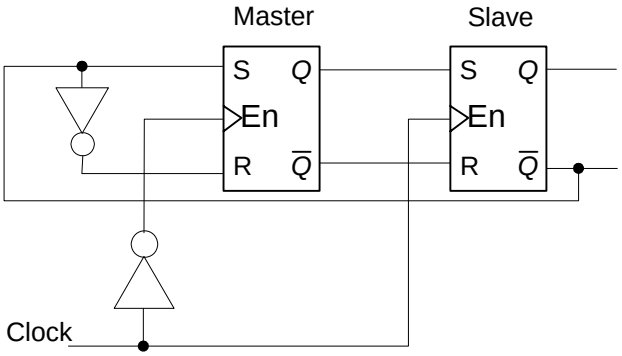


Quando ligamos a entrada D à saída $\overline{Q}$, temos um circuito semelhante ao flip-flop JK quando J=K=1.

Neste caso temos:

S = $\overline{Q}$ e R = Q

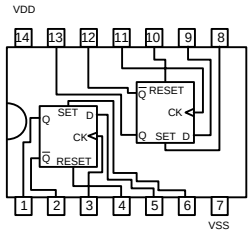




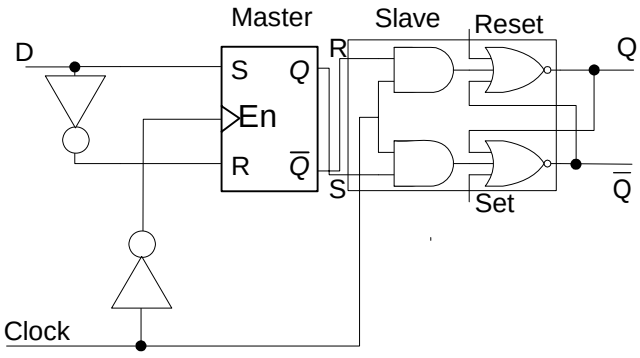
Um circuito integrado muito usado é o 4013 da família CMOS.

Este componente tem dois flip-flop JK com duas entradas Set e Reset. Estas entradas não estão sincronizadas com a entrada de clock, desta forma o Set e o Reset podem alterar a saída Q em qualquer momento diferente do clock.

4013



Para que as entradas funcionem num modo assíncrono, independentes do clock, elas têm que estar ligadas na saída do flip-flop, conforme o esquema:

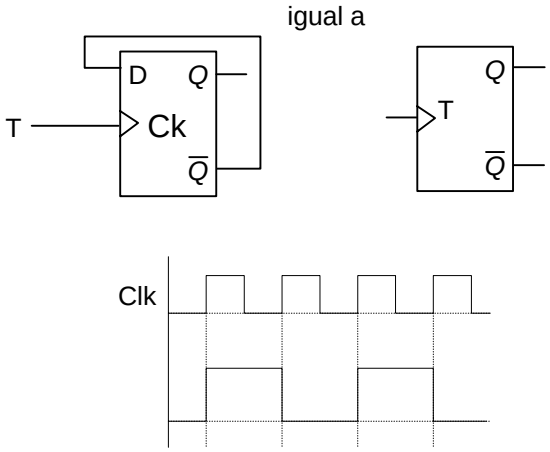


FLIP FLOP T

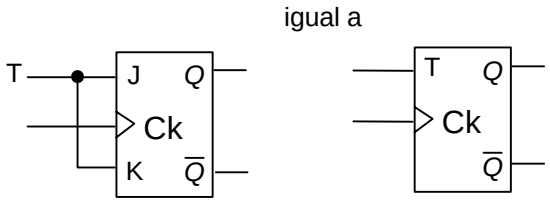
O flip-flop T (Toggle) pode ter uma ou duas entradas e pode ser construído com um flip-flop D ou JK master slave. O próprio nome indica o seu modo de funcionamento, quer dizer; o próximo estado é sempre o contrário do estado actual.

Este flip-flop é usado normalmente em contadores ou divisores de frequência.

Ligação de um flip-flop D para funcionar como flip-flop T:

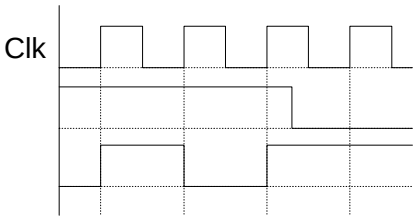


Ligação de um flip-flop JK para funcionar como flip-flop T:



Quando a entrada T=0, então J=K=0, esta combinação corresponde na tabela de verdade do JK à condição "mantém o estado anterior".

Quando a entrada T=1, então J=K=1, esta combinação corresponde na tabela de verdade do JK à condição "inverte".



CONTADORES ASSÍNCRONOS

Um contador é um circuito sequencial, de aplicação geral cujas saídas representam num determinado código, o número de impulsos que se aplicam à entrada.

São constituídos por vários flip-flops T ligados entre si, de maneira a que as suas saídas mudem de estado, quando se aplicam impulsos na entrada.

Quando o contador atinge o valor máximo da sua capacidade, começa a contar de novo desde o zero, no impulso de clock imediato.

Dependendo da forma de operação, os contadores podem ser ascendentes, se a contagem aumenta em cada impulso, descendentes, se a contagem diminui ou, ascendentes e descendentes (up-down counters).

Por outro lado, os contadores dividem-se em assíncronos e síncronos. Os contadores síncronos são descritos mais adiante.

Nos contadores assíncronos, o sinal de clock aplica-se à entrada do primeiro flip-flop, a saída deste à entrada do próximo e, assim sucessivamente. O tempo de propagação deste tipo de contadores é superior ao dos contadores síncronos.

Existem, também, contadores binários e decimais (contadores de décadas), assim o número de estados possíveis nas saídas seja de 2^n ou de 10. Outro tipo de contadores pode ser referenciado através do seu módulo. Assim, um contador cujo número de estados nas saídas seja seis (0, 1, 2, 3, 4, 5) será um contador módulo 6.

CONTADORES CRESCENTES

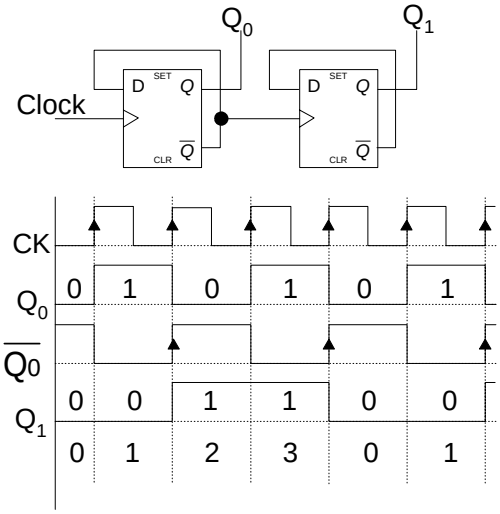
O contador crescente é caracterizado por ter o clock ligado à saída $\overline{Q}$ do fli-flop anterior. A saída Q_0 tem o dobro do período do clock, e a saída Q_1 tem o do período de Q_0. Assim a saída Q1 tem $\frac{1}{4}$ da frequência do clock.

O módulo de contagem deste contador é 4, e é calculado da seguinte forma:

Módulo = $2^n \rightarrow$ n é o número de flip-flops.

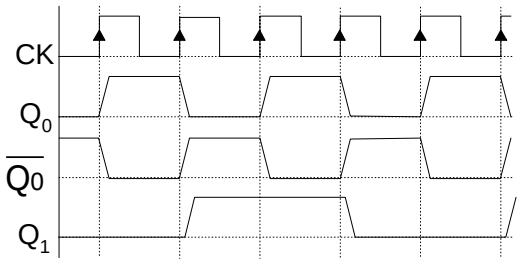
O módulo de contagem também é a relação de frequência entre o clock e o ultimo Q.

Neste caso é: Módulo = $\frac{\text{Clock}}{Q_1}$



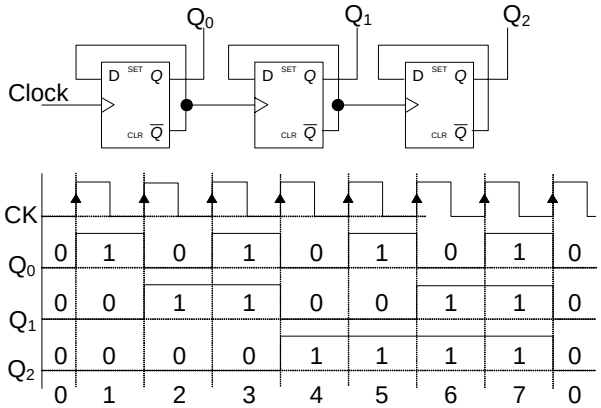
A saída Q1 depende da transição ascendente de $\overline{Q_0}$, e equivale à transição descendente de Q_0.

Devido ao tempo de propagação e mudança de estado, há um pequeno atraso desde a transição de clock até à mudança de estado nas saídas Q.



Podemos observar no diagrama temporal que há um atraso do clock para Q_0 e um atraso maior para Q_1. Este atraso vai aumentando à medida que vamos acrescentando mais flip-flops ao contador.

O contador crescente módulo 8 necessita 3 flip-flops. A ligação entre o flip-flops é igual ao circuito anterior.

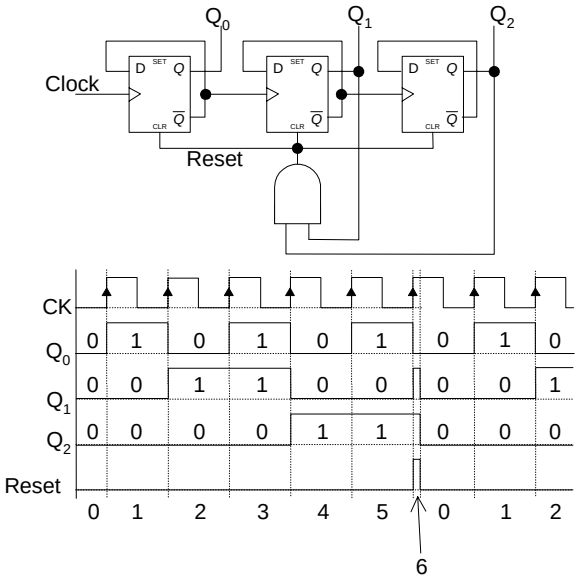


A relação de frequência entre o Clock e Q_2 é 8.

Se for necessário construir um contador com módulo diferente de 2n, é necessário recorrer a uma lógica combinatória para que ocorra um reset, de modo que o contador recomece a contagem no ponto desejado.

No circuito seguinte está representado um contador de módulo 6. Este contador deve contar 0, 1, 2, 3, 4, 5 e depois regressar a 0.

Tabela de verdade				
Q_2	Q_1	Q_0	Decimal	Reset
0	0	0	0	
0	0	1	1	
0	1	0	2	
0	1	1	3	
1	0	0	4	
1	0	1	5	
1	1	0	6	$Q_2.Q_1$

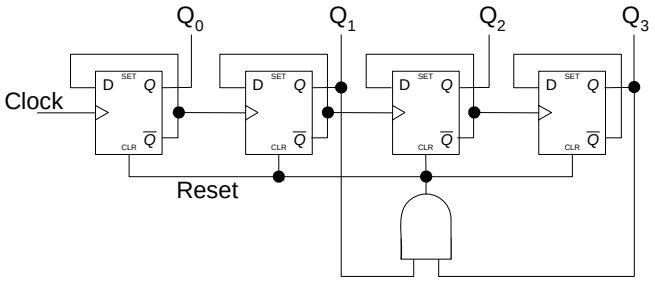


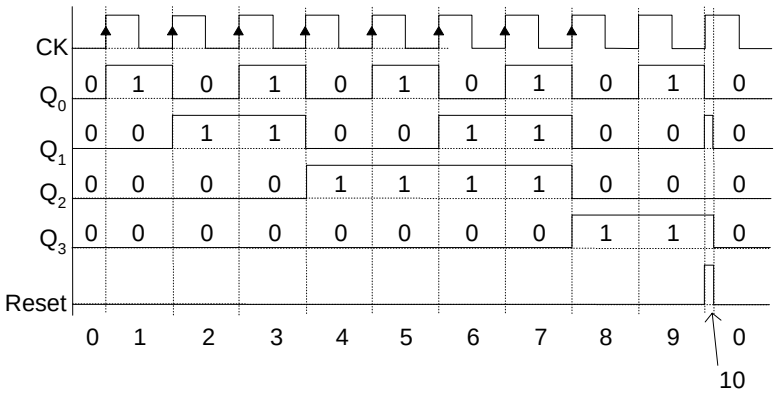
A combinação referente ao número 6 aparece num curto espaço de tempo (ver diagrama), este é o tempo necessário para que o circuito lógico detecte este valor e accione o Reset.

O contador módulo 10, ou contador decimal (BCD) utiliza o mesmo método. Utilizamos 4 flip-flops que permitem a contagem de 0 a 15 (módulo 16), mas limitamos a contagem com um circuito lógico combinatório.

Vamos analisar a sua tabela de verdade e vamos comparar com o diagrama temporal. No diagrama temporal, podemos ver que tal como o circuito anterior, este contador também tem um tempo de Reset bastante curto, que é o tempo necessário para o circuito combinatório responder.

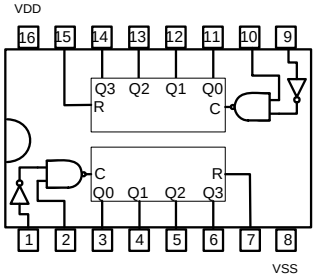
Tabela de verdade					
Q_3	Q_2	Q_1	Q_0	Decimal	Reset
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	2	
0	0	1	1	3	
0	1	0	0	4	
0	1	0	1	5	
0	1	1	0	6	
0	1	1	1	7	
1	0	0	0	8	
1	0	0	1	9	
1	0	1	0	10	$Q_3.Q_1$





Existe no mercado vários circuitos integrados que desempenham esta função, sendo um deles o 4518 da família CMOS. Este circuito integrado contém dois contadores. Cada contador tem uma entrada de Reset (pino 7 e 15) e uma entrada de clock (pino 2 e 10) com enable (pino 1 e 9). O modelo 4518 é contador módulo 10 e o 4520 é contador módulo 16.

4518/4520

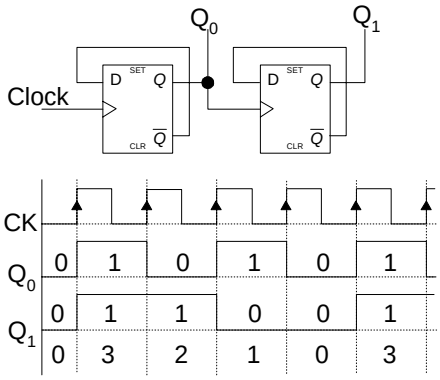


CONTADORES DECRESCENTES

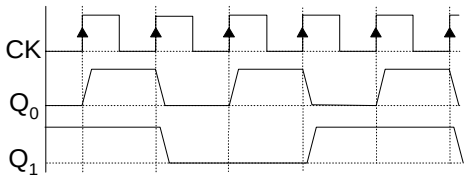
O contador decrescente difere do crescente no modo que é ligado o clock. Neste caso o clock é ligado à saída Q enquanto que o crescente é ligado à saída $\overline{Q}$.

O módulo de contagem é efectuado da mesma forma: 2^n

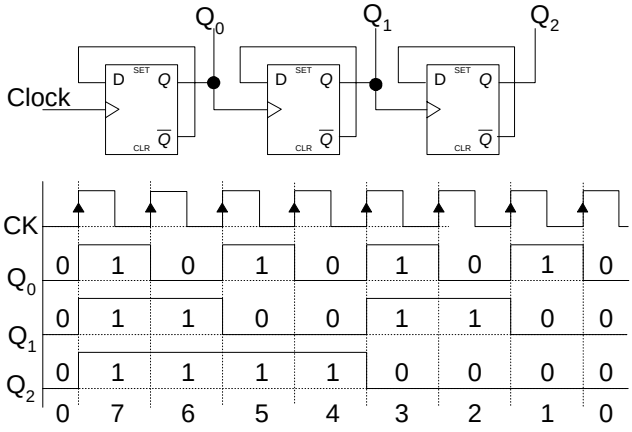
Ou pode também ser calculado pela relação entre a frequência de clock e $\overline{Q_1}$: $\frac{\text{Clock}}{Q_1}$



Neste contador também existem tempos de atraso iguais ao contador crescente.



Se for necessário aumentar o módulo do contador, basta aumentar o número de flip-flops de acordo com a regra 2^n.

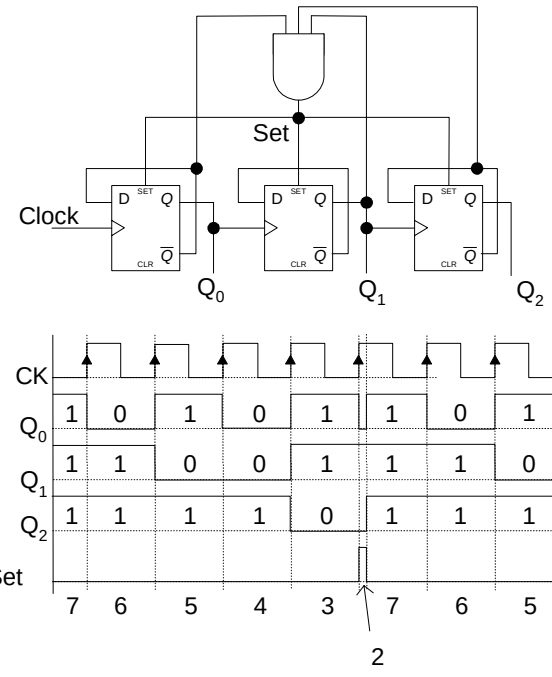


É possível também neste caso estabelecer o início e fim de contagem do mesmo modo que foi feito no contador crescente módulo 6 e módulo 10.

Vamos ver um exemplo de um contador que efectua a seguinte contagem: 7, 6, 5, 4, 3, 7, 6, 5, ….

Para este efeito vamos analisar a tabela de verdade e comparar com o diagrama temporal.

Tabela de verdade				
Q_2	Q_1	Q_0	Decimal	Set
1	1	1	7	
1	1	0	6	
1	0	1	5	
1	0	0	4	
0	1	1	3	
0	1	0	2	$\overline{Q_2}.Q_1.\overline{Q_0}$

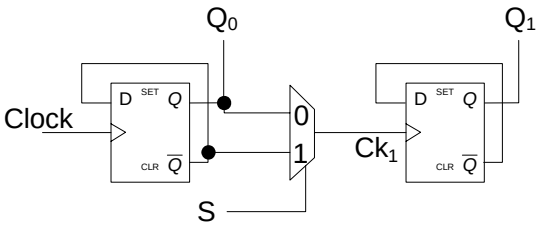


A combinação referente ao número 2 ocorre num instante de tempo muito curto, tal como ocorria no contador crescente módulo 6.

CONTADORES CRESCENTES/DECRESCENTES

Nos circuitos anteriores podemos verificar que o contador crescente tem a entrada de clock ligada à saída $\overline{Q}$ e o contador decrescente tem a entrada de clock ligada à saída Q.

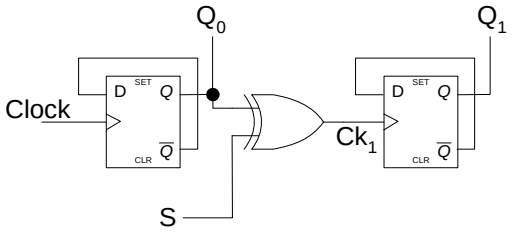
Podemos então usar um multiplexer para efectuar a mudança de crescente para decrescente, mudando o ponto onde o clock está ligado.



Quando S=0 temos a saída Q ligada ao clock, quando S=1 temos a saída $\overline{Q}$ ligada ao clock.

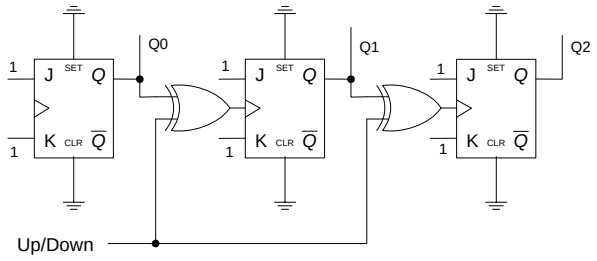
A expressão lógica de um multiplexer é $\overline{S}$. A + S . B, podemos substituir a letra A pela letra Q, e a letra B pela letra $\overline{Q}$. Assim obtemos a expressão: $\overline{S}$. Q + S . $\overline{Q}$ = S $\oplus$ Q e o respectivo circuito.

O multiplexer usado no circuito anterior pode ser substituído pela função XOR conforme ficou comprovado.



Este contador é de módulo 4, e permite efectuar a contagem 0, 1, 2, 3 quando S=1, ou a contagem 3, 2, 1, 0 quando S=0.

É possível construir um contador crescente/decrescente com um módulo maior que 4, para isso basta adicionar os flip-flops necessários e colocar um XOR entre a saída Q e a entrada de clock.



Este circuito tem a capacidade de efectuar a contagem de 0 a 7 quando S=1, ou de 7 a 0

quando S=0.

CONTADORES SÍNCRONOS

Os contadores síncronos são aqueles em que os impulsos de relógio se aplicam simultaneamente a todos os flip-flops e portanto, as suas saídas variam ao mesmo tempo.

Estes contadores podem usar flip-flops JK ou tipo D. A ligação dos flip-flops deve ser de acordo com sequência obtida a partir da tabela de verdade.

CONTADORES CRESCENTES

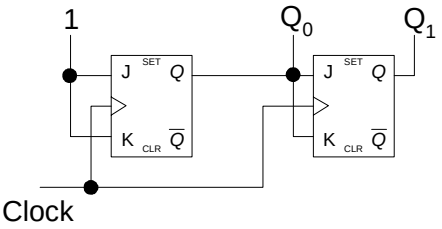
Um contador de módulo 4 é fácil de construir, para isso vamos recorrer a uma tabela de verdade:

Contador síncrono crescente módulo 4				
JK_1	Q_1	JK_0	Q_0	Decimal
0	0	1	0	0
1	0	1	1	1
0	1	1	0	2
1	1	1	1	3
0	0	1	0	0

A saída Q_0 muda sempre de estado em cada clock, por isso o JK_0 é sempre igual a 1, de acordo com a tabela de verdade do flip-flop JK:

Tabela de verdade do flip-flop JK		
J	K	Condição
0	0	Mantém
0	1	Reset
1	0	Set
1	1	Inverte

A saída Q_1 inverte apenas quando Q_0=1, por isso a entrada JK_1 é igual a Q_0. Assim obtemos o seguinte circuito.



A tabela de verdade também pode ser construída da seguinte forma:

Contador síncrono crescente módulo 4						
J_1	K_1	Q_1	J_0	K_0	Q_0	Decimal
0	X	0	1	X	0	0
1	X	0	X	1	1	1
X	0	1	1	X	0	2
X	1	1	X	1	1	3
0	X	0	1	X	0	0

A letra “X” significa que é indiferente termos nível lógico “0” ou nível lógico 1, por isso as entradas J_0 e K_0 podem estar sempre ligadas a “1”.

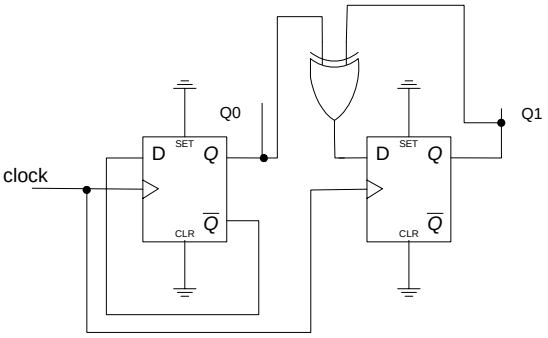
As entradas J_1 e K_1 podem ser unidas porque há sempre uma delas que tem a letra “X”, isso significa que é indiferente o seu nível lógico.

Também é possível construir um contador síncrono com flip-flops tipo D de acordo com a seguinte tabela de verdade:

Contador síncrono módulo 4				
D_1	Q_1	D_0	Q_0	Decimal
0	0	1	0	0
1	0	0	1	1
1	1	1	0	2
0	1	0	1	3
0	0	1	0	0

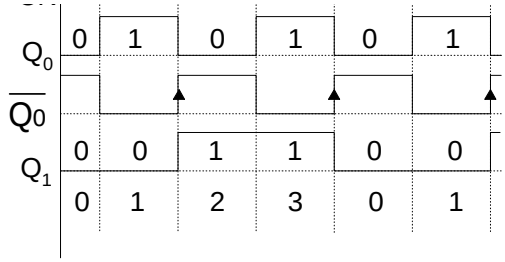
$D_0 = \overline{Q_0}$

$D_1 = \overline{Q_1}.Q_0 + \overline{Q_0}.Q_1$

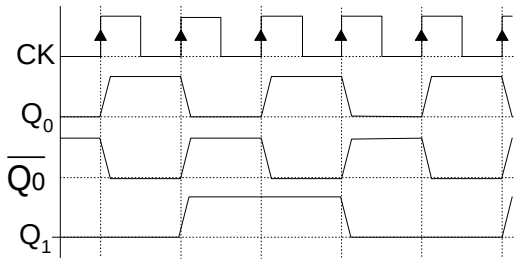


$D_1 = Q_0 \oplus Q_1$

O diagrama temporal do contador síncrono é aparentemente igual ao do contador assíncrono.



As diferenças aparecem quando ampliamos a imagem de modo a ser possível observar o tempo de transição do nível lógico 0 para 1 e vice-versa.



Neste tipo de contador, só há atraso entre o clock e as saídas Q, mas todas as saídas Q mudam de estado em simultâneo. Já não existe a acumulação de tempo da saída Q_0 para a saída Q_1.

Vamos agora ver o método de construção de um contador crescente módulo 8.

Contador crescente módulo 8						
JK_2	Q_2	JK_1	Q_1	JK_0	Q_0	Decimal
0	0	0	0	1	0	0
0	0	1	0	1	1	1
0	0	0	1	1	0	2
1	0	1	1	1	1	3
0	1	0	0	1	0	4
0	1	1	0	1	1	5
0	1	0	1	1	0	6

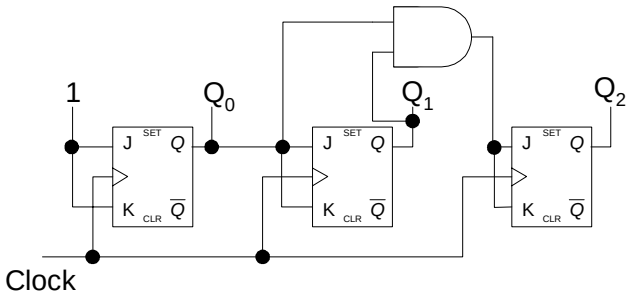
1	1	1	1	1	1	7
0	0	0	0	1	0	0

$JK_0 = 1$

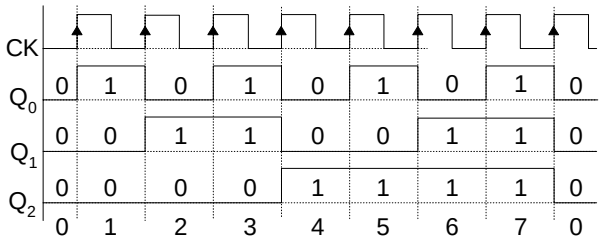
$JK_1 = Q_0$

$JK_2 = \overline{Q_2}.Q_1.Q_0 + Q_2.Q_1.Q_0$

$JK_2 = (\overline{Q_2} + Q_2).Q_1.Q_0 = Q_1.Q_0$



Tal como o contador módulo 4, o contador módulo 8 também tem um diagrama semelhante ao contador assíncrono, mas com a vantagem de não haver acumulação de atrasos entre as várias saídas Q.



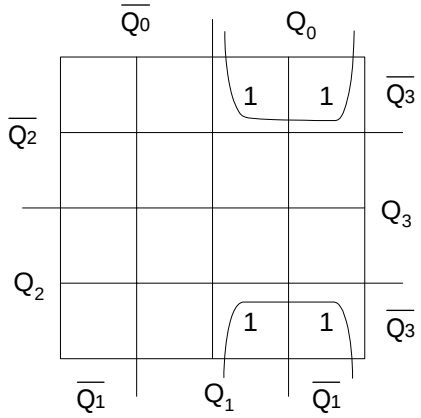
Contador Decimal síncrono:

Contador BCD síncrono								
JK_3	Q_3	JK_2	Q_2	JK_1	Q_1	JK_0	Q_0	Decimal
0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	1	1	1
0	0	0	0	0	1	1	0	2
0	0	1	0	1	1	1	1	3
0	0	0	1	0	0	1	0	4
0	0	0	1	1	0	1	1	5
0	0	0	1	0	1	1	0	6

1	0	1	1	1	1	1	1	7
0	1	0	0	0	0	1	0	8
1	1	0	0	0	0	1	1	9
0	0	0	0	0	0	1	0	0

$JK_0 = 1$

$JK_1 = \overline{Q_3} . \overline{Q_2} . \overline{Q_1} . Q_0 + \overline{Q_3} . \overline{Q_2} . Q_1 . Q_0 + \overline{Q_3} . Q_2 . \overline{Q_1} . Q_0 + \overline{Q_3} . Q_2 . Q_1 . Q_0$



$JK_1 = \overline{Q_3} . Q_0$

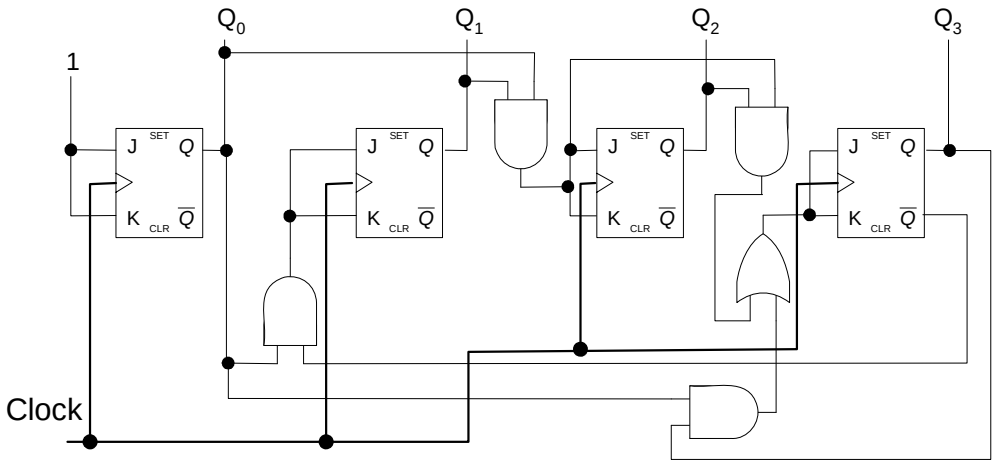
$JK_2 = \overline{Q_3} . \overline{Q_2} . Q_1 . Q_0 + \overline{Q_3} . Q_2 . Q_1 . Q_0 = \overline{Q_3} . Q_1 . Q_0 . (\overline{Q_2} + Q_2) = \overline{Q_3} . Q_1 . Q_0$

A condição Q_1=1 e Q_0=1 só ocorre nesta tabela quando Q_3=0, por isso podemos desprezar a variável $\overline{Q_3}$.
Assim $JK_2 = Q_1 . Q_0$

$JK_3 = \overline{Q_3} . Q_2 . Q_1 . Q_0 + Q_3 . \overline{Q_2} . \overline{Q_1} . Q_0$

Podemos simplificar esta expressão ($Q_3 . \overline{Q_2} . \overline{Q_1} . Q_0$) do seguinte modo: $Q_3 . Q_0$ porque esta condição só ocorre uma vez nesta tabela.

$JK_3 = \overline{Q_3} . Q_2 . Q_1 . Q_0 + Q_3 . Q_0 = Q_2 . Q_1 . Q_0 + Q_3 . Q_0 = (Q_2 . Q_1 + Q_3) . Q_0$



CONTADORES DECRESCENTES

A mesma regra que foi aplicada nos contadores crescentes, também é aplicada nos contadores decrescentes:

Contador síncrono decrescente módulo 4				
JK_1	Q_1	JK_0	Q_0	Decimal
0	1	1	1	3
1	1	1	0	2
0	0	1	1	1
1	0	1	0	0
0	1	1	1	3

A saída Q_0 muda sempre de estado em cada clock, por isso o JK_0 é sempre igual a 1, de acordo com a tabela de verdade do flip-flop JK:

Tabela de verdade do flip-flop JK		
J	K	Condição
0	0	Mantém
0	1	Reset
1	0	Set
1	1	Inverte

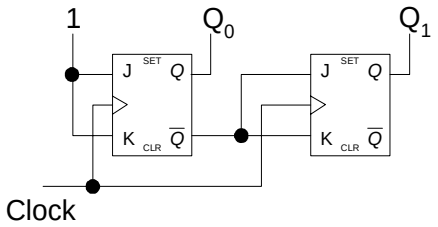
A saída Q_1 inverte apenas quando Q_0=0, por isso a entrada JK_1 é igual a $\overline{Q_0}$. Assim obtemos o seguinte circuito.

A tabela de verdade também pode ser construída da seguinte forma:

Contador síncrono decrescente módulo 4						
J_1	K_1	Q_1	J_0	K_0	Q_0	Decimal
X	0	1	X	1	1	3
X	1	1	1	X	0	2
0	X	0	X	1	1	1
1	X	0	1	X	0	0
X	0	1	X	1	1	3

A letra “X” significa que é indiferente termos nível lógico “0” ou nível lógico 1, por isso as entradas J_0 e K_0 podem estar sempre ligadas a “1”.

As entradas J_1 e K_1 podem ser unidas porque há sempre uma delas que tem a letra “X”, isso significa que é indiferente o seu nível lógico.



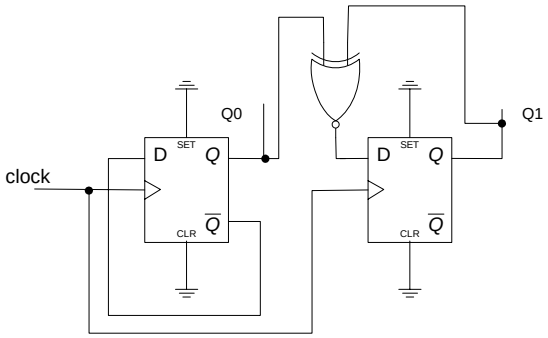
Também é possível construir um contador síncrono com flip-flops tipo D de acordo com a seguinte tabela de verdade:

Contador síncrono módulo 4				
D_1	Q_1	D_0	Q_0	Decimal
1	1	0	1	3
0	1	1	0	2
0	0	0	1	1
1	0	1	0	0
1	1	0	1	3

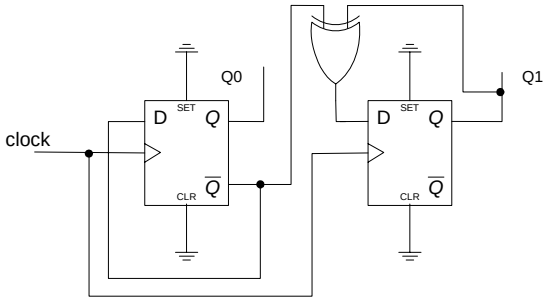
$D_0 = \overline{Q_0}$

$D_1 = \overline{Q_1} . \overline{Q_0} + Q_1.Q_0$

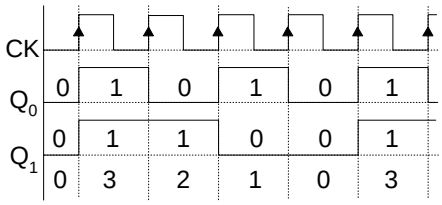
$D_1 = \overline{Q_1 \oplus Q_0}$



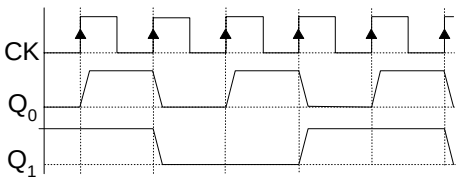
Como já foi visto anteriormente $\overline{Q_1 \oplus Q_0} = \overline{Q_0} \oplus Q_1$, então podemos alterar o circuito:



Neste circuito também temos um diagrama temporal semelhante ao contador assíncrono.



Temos a mesma vantagem do contador síncrono crescente, porque não há acumulação de atrasos entre as várias saídas Q. Apenas existe atraso entre as saídas e o clock.



Vamos agora ver o método de construção de um contador decrescente módulo 8.

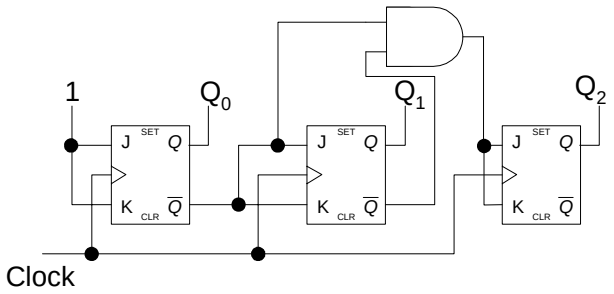
Contador decrescente módulo 8						
JK_2	Q_2	JK_1	Q_1	JK_0	Q_0	Decimal
0	1	0	1	1	1	7
0	1	1	1	1	0	6
0	1	0	0	1	1	5
1	1	1	0	1	0	4
0	0	0	1	1	1	3
0	0	1	1	1	0	2
0	0	0	0	1	1	1
1	0	1	0	1	0	0
0	1	0	1	1	1	7

$JK_0 = 1$

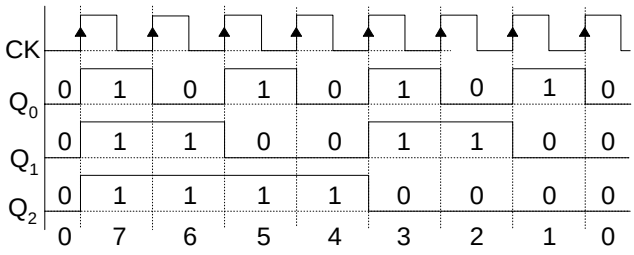
$JK_1 = \overline{Q_0}$

$JK_2 = \overline{Q_2} \cdot \overline{Q_1} \cdot \overline{Q_0} + Q_2 \cdot \overline{Q_1} \cdot \overline{Q_0}$

$JK_2 = (\overline{Q_2} + Q_2) \cdot \overline{Q_1} \cdot \overline{Q_0} = \overline{Q_1} \cdot \overline{Q_0}$

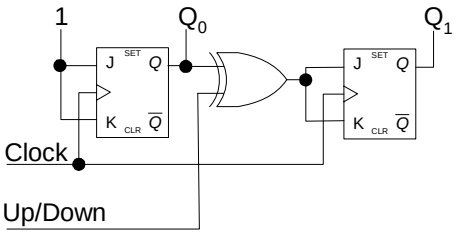


O diagrama temporal está de acordo com a sua sequência de contagem (7 a 0), igual ao contador assíncrono, mas sem haver acumulação de atrasos entre as várias saídas Q.

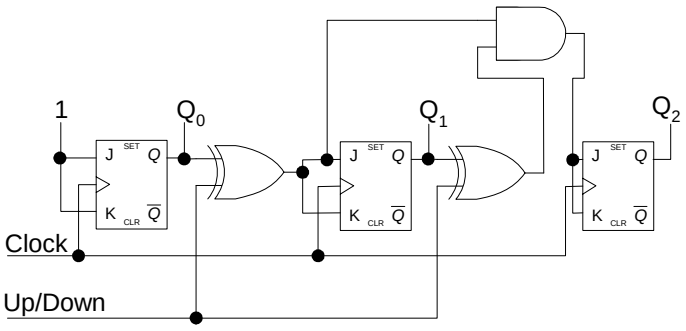


CONTADORES CRESCENTES/DECRESCENTES

A diferença entre um contador crescente e um contador decrescente é o ponto de ligação das entradas JK. No contador crescente, as entradas JK ligam nas saídas Q e no contador decrescente ligam nas saídas $\overline{Q}$. Assim podemos utilizar o mesmo método dos contadores assíncronos, utilizando uma porta lógica XOR.

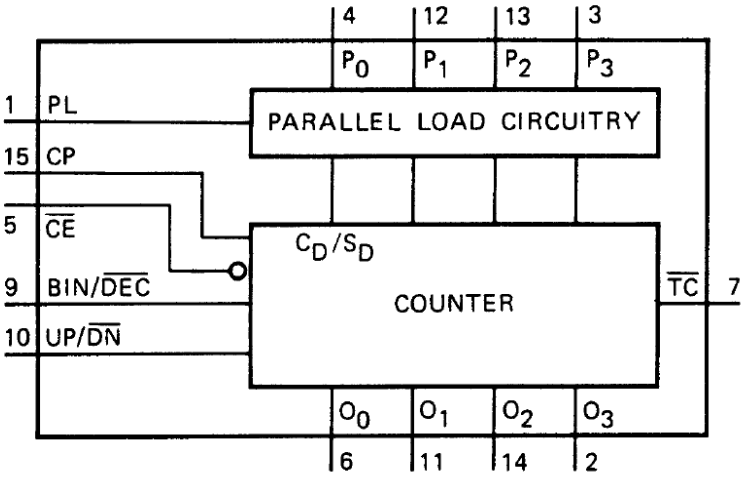


No contador módulo 8, podemos aplicar a mesma regra ao circuito, mas neste caso temos que usar duas portas XOR.



O circuito integrado 4029 é um contador síncrono crescente/decrescente, com uma entrada de selecção BIN/$\overline{DEC}$. Quando esta entrada tem nível lógico 0, é um contador BCD, quando esta entrada tem nível lógico 1 é um contador binário. Tem também uma entrada UP/$\overline{DN}$. Quando esta entrada tem nível lógico 0, é um contador decrescente, quando esta entrada tem nível lógico 1 é um contador crescente.

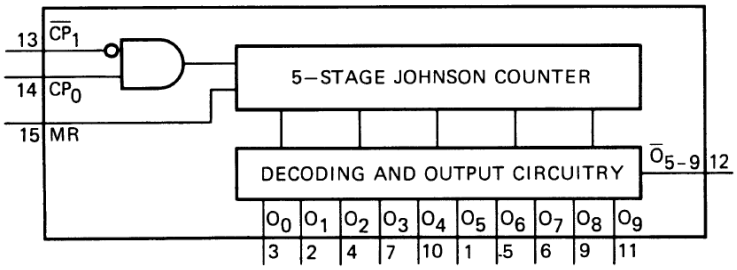
A entrada $\overline{CE}$ é activada a 0, e é o enable do clock.



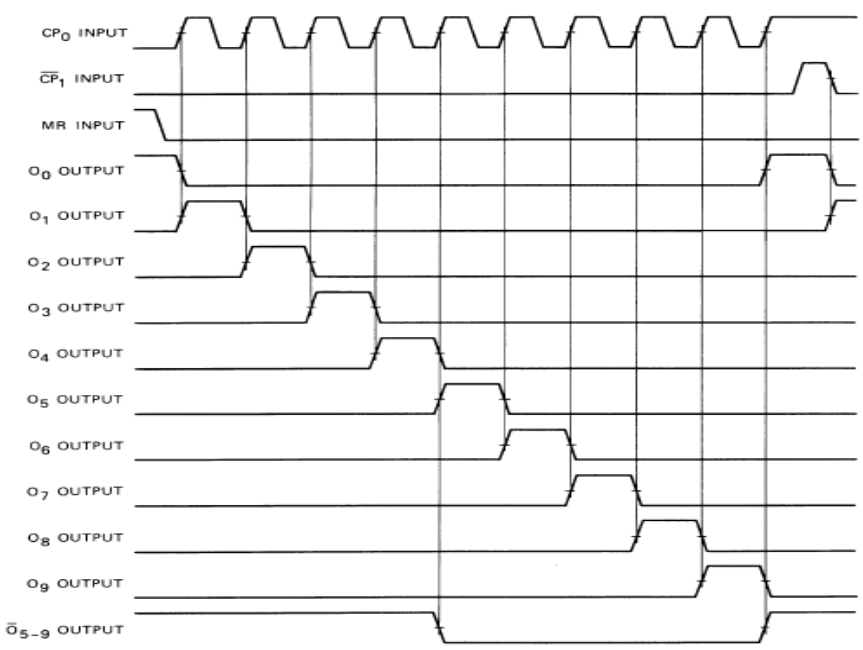
Existem outras entradas que permitem entrada de dados em paralelo (PL, P_0, P_1, P_2, P_3)

Tabela de verdade					
PL	BIN/$\overline{\text{DEC}}$	UP/$\overline{\text{DN}}$	$\overline{\text{CE}}$	CP	Modo
1	X	X	X	X	Carga paralela
0	X	X	H	X	Não muda
0	0	0	0	↑	Decimal, decrescente
0	0	1	0	↑	Decimal, crescente
0	1	0	0	↑	Binário, decrescente
0	1	1	0	↑	Binário, crescente

O Circuito integrado 4017, é um contador decimal ligado a descodificador decimal, com dez saídas, desde O_0 até O_9. A entrada de clock CP_0 é utilizada para incrementar o contador em cada transição de clock ascendente. A entrada de clock $\overline{CP_1}$ é utilizada para incrementar o contador em cada transição de clock descendente. A entrada $\overline{CP_1}$ também pode usada como entrada $\overline{EN}$.

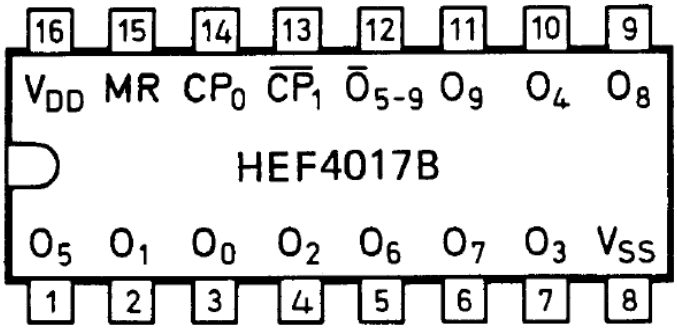


O diagrama temporal deste contador é o seguinte:



A tabela de verdade e o pinout do 4017 estão representados a seguir:

Tabela de verdade			
MR	CP_0	$\overline{CP_1}$	Operação
1	X	X	Todas as saídas a 0
0	H	↓	Contador avança
0	↑	0	Contador avança
0	0	X	Não muda
0	X	1	Não muda
0	1	↑	Não muda
0	↓	0	Não muda


16 15 14 13 12 11 10 9
VDD MR CP0 CP1 O5-9 O9 O4 O8
HEF4017B
O5 O1 O0 O2 O6 O7 O3 VSS
1 2 3 4 5 6 7 8

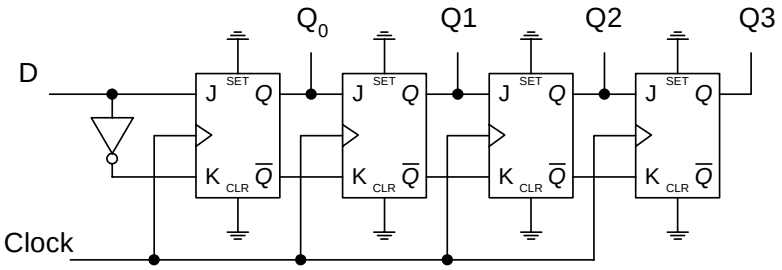
REGISTOS DE DESLOCAMENTO

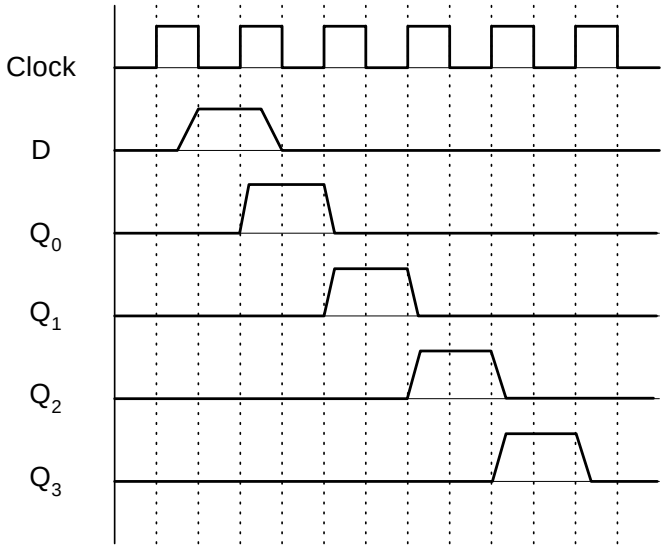
Os registos de deslocamento podem possuir uma combinação de entradas e saídas série, e paralelas, incluindo a possibilidade de configurar as entradas e saídas.

ACTIVAÇÃO SEQUENCIAL E DIAGRAMA TEMPORAL

O esquema seguinte representa o modelo mais simples de registo de deslocamento. Os dados são colocados na entrada D e são deslocados uma posição para a direita em cada ciclo de clock.

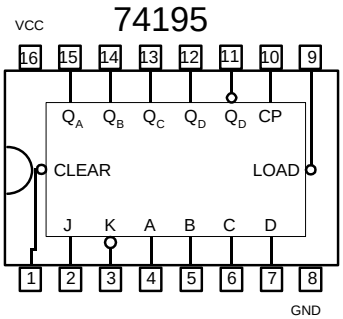
Os dados são armazenados em cada flip-flop, na saída 'Q', de modo que existem quatro "espaços" para armazenamento disponíveis nesta configuração, sendo desta forma um registo de deslocamento de 4 bits. Esta configuração realiza uma leitura destrutiva, visto que os dados são perdidos ao serem deslocados para a extrema direita.





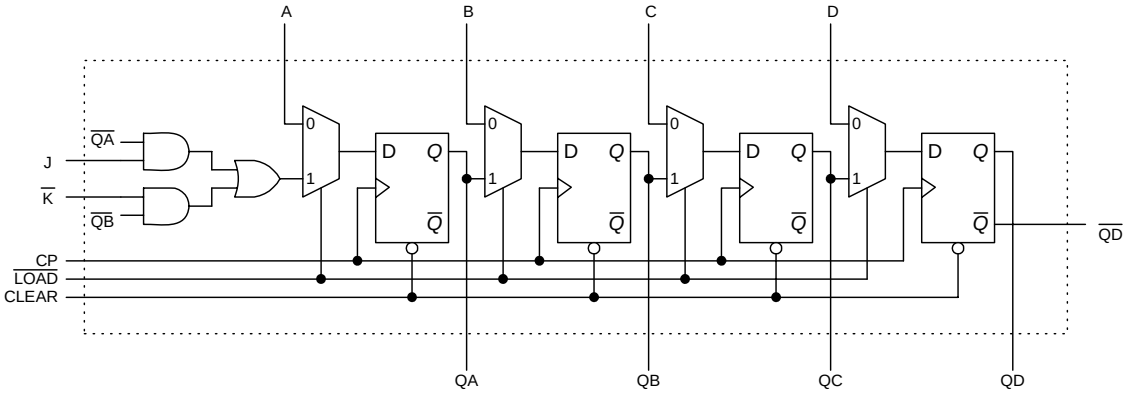
TIPOS DE REGISTOS DE DESLOCAMENTO E CIRCUITOS INTEGRADO

O circuito integrado 74195 é um registo de deslocamento de 4 bits de alta velocidade, podendo atingir a frequência de 50MHz. É utilizado numa grande variedade de aplicações com registos e contadores.



Pinos	Descrição	Activado
$\overline{PE}$	Enable das entradas paralelas	Zero
P0 a P3	Entradas de dados em paralelo	Um
J	Entrada J do primeiro estágio	Um
$\overline{K}$	Entrada K do primeiro estágio	Zero
CP	Clock Pulse	Um
Q0 a Q3	Saídas em paralelo	Um
$\overline{Q3}$	Saída complementada do ultimo estágio	Zero

O circuito lógico e a tabela de verdade mostram o funcionamento do circuito 74195.



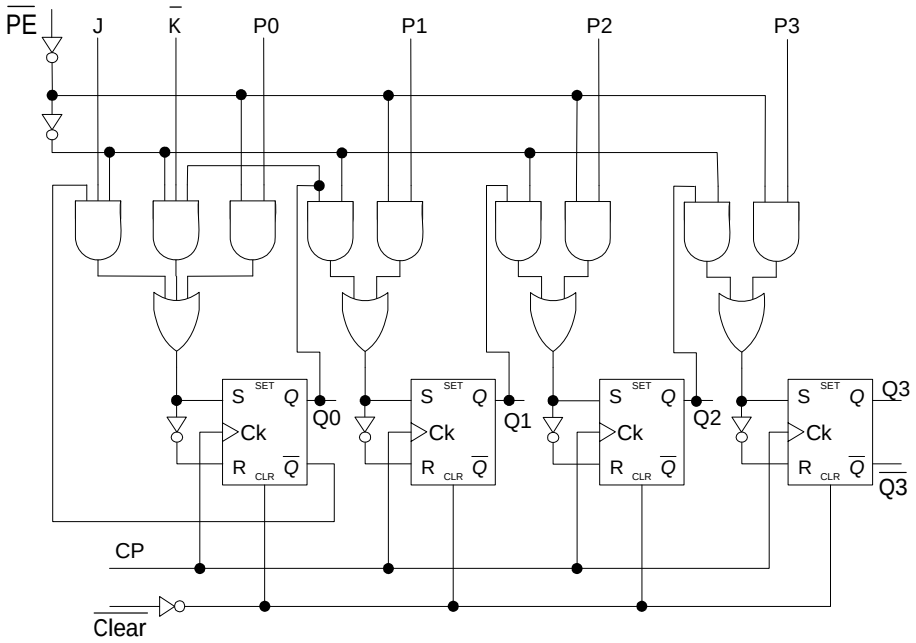
O 74195, tem dois modos de funcionamento primários, deslocamento à direita e carga em paralelo que são controlados pela entrada $\overline{PE}$. Quando $\overline{PE}=1$, os dados entram em série pelo primeiro flip-flop através das entradas J e $\overline{K}$ e deslocados pelas saídas Q0, Q1, Q2 e Q3 em cada transição de clock. Quando $\overline{PE}=0$, o 74195 funciona como quatro flip-flops tipo D. Os dados entram em paralelo pelas entradas P0, P1, P2 e P3 e

são transferidos paras as respectivas saídas Q0, Q1, Q2 e Q3 na transição de clock ascendente. É possível fazer a rotação à esquerda, ligando as saídas Qn às entradas Pn-1 e colocando $\overline{PE}$=0.

Modos	Entradas					Saídas				
	$\overline{CLEAR}$	$\overline{PE}$	J	$\overline{K}$	Pn	Q0	Q1	Q2	Q3	$\overline{Q3}$
Reset	0	X	X	X	X	0	0	0	0	1
Liga o primeiro estágio	1	1	1	1	X	1	Q0	Q1	Q2	$\overline{Q2}$
Desliga o primeiro estágio	1	1	0	0	X	0	Q0	Q1	Q2	$\overline{Q2}$
Inverte o primeiro estágio	1	1	0	1	X	$\overline{Q0}$	Q0	Q1	Q2	$\overline{Q2}$
Mantém o primeiro estágio	1	1	1	0	X	Q0	Q0	Q1	Q2	$\overline{Q2}$
Carga em paralelo	1	0	X	X	Pn	P0	P1	P2	P3	$\overline{P3}$

Todas as transferências de dados série ou paralelas são síncronas, e ocorrem na transição ascendente de clock.

O flip-flop D é constituído por flip-flop RS e um inversor nas entradas S e R.



ARQUITECTURA BÁSICA DOS COMPUTADORES

TERMINOLOGIA

- Bit

Um digito binário pode representar 0 ou 1 e é normalmente chamado de bit.

- Byte

Um conjunto de oito bits é chamado de Byte

- Software

Refere-se aos programas escritos para computador

- Hardware

É o nome dado aos dispositivos físicos e circuitos do computador

- CPU (Central Processing Unit)

O CPU controla as operações de um computador. O CPU lê as instruções que estão na memória, descodifica-as numa série de acções, e leva a cabo estas acções numa série de passos.

O CPU também contem o contador de endereços ou o registo de ponteiro de instrução, que contém o endereço da próxima instrução que vai ser lida da memória. Também tem registos de uso geral para armazenar dados em binário; e todo o circuito de controlo que gera os sinais para os barramentos de endereço e dados.

- IC (Integrated Circuits)

Nome dado a um circuito integrado. Componente constituído por transístores e outros componentes com uma determinada função.

- Bus de endereços

Este bus consiste em 16, 20, 24 ou 32 linhas de sinal em paralelo. Nestas linhas o CPU envia o endereço de memória onde se vai ler ou escrever.

O número de endereços que o CPU pode endereçar é determinado pelo número de linhas de endereço. Se o CPU tem n linhas de endereço, então pode endereçar 2^n posições de memória.

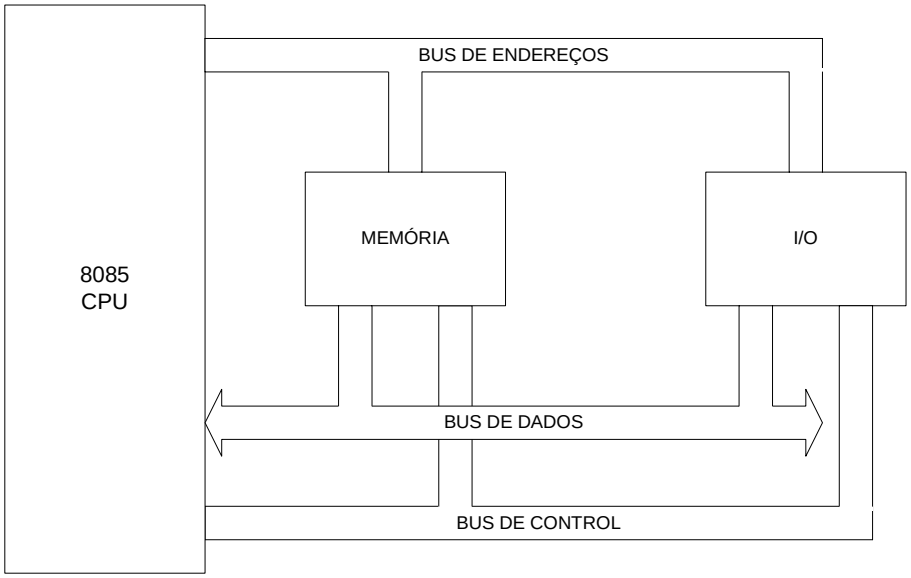
- Bus de dados

O bus de dados consiste em 8, 16, 32, ou 64 linhas de sinal em paralelo. Através destas linhas o CPU consegue ler ou escrever dados da memória ou de uma entrada/saída. Há muitos dispositivos ligados ao bus de dados, mas apenas uma das suas saída está activada de cada vez.

LAYOUT DE UM COMPUTADOR

O microprocessador 8085 pode aceder a 65536 posições individuais de memória numa gama de 0 a 65535 (0 a FFFFHex), 64 KBytes de memória. Existem dois tipos de memória, ROM (Read Only Memory) e RAM (Random Access Memory). Ambas as memórias têm linhas de endereço e dados ligados ao microprocessador. Estas ligações são feitas através do Bus de dados e Bus de endereços, o microprocessador usa-os para ler ou escrever dados em memória.

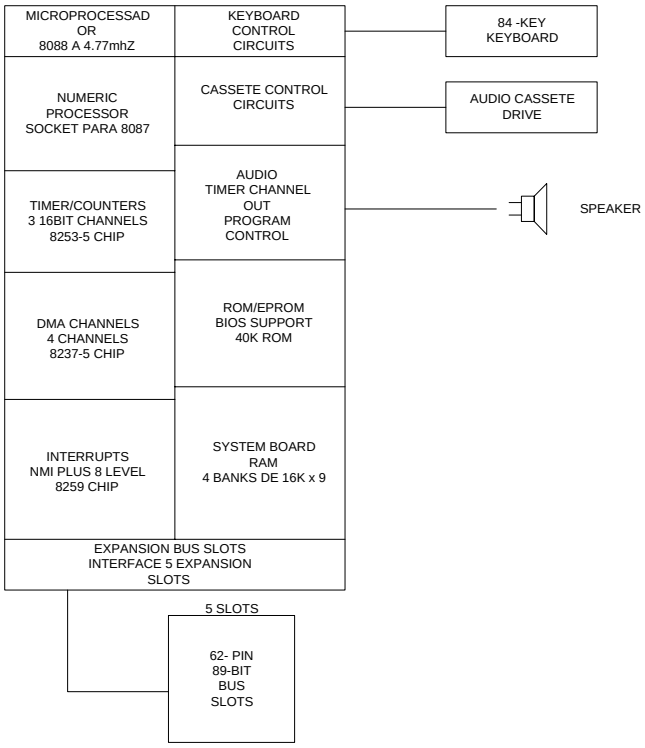
Os circuitos no 8085 usam 8 bits no Bus de dados e 16 bits no Bus de endereços. Assim qualquer valor entre 0 e FF Hex pode ser lido ou escrito em memória RAM e apenas lido na ROM.



O 8085 pode enviar ou receber dados de outros circuitos para além da memória. Quando o microprocessador quer comunicar com uma entrada ou saída, “desliga” a memória, liga a entrada ou saída e envia o endereço através do Bus de endereços nos 8 bits menos significativos (A0 a A7) e nos bits mais significativos simultaneamente (A8 a A15). Visto que o endereçamento de I/O é de 8 bits, apenas as posições entre 0 e FF Hex podem ser seleccionadas.

O Bus de controlo é um conjunto de ligações que controlam a escrita e a leitura em memória e dispositivos I/O .

O PC original era baseado numa board com 32K de RAM e cinco slots de expansão. O PC suportava um ou dois floppy's, com a capacidade de 160K cada. Logo após esta introdução no mercado, a memória foi aumentada para 64K. O PC original usava o microprocessador Intel 8088 trabalhando a 4.77MHz.

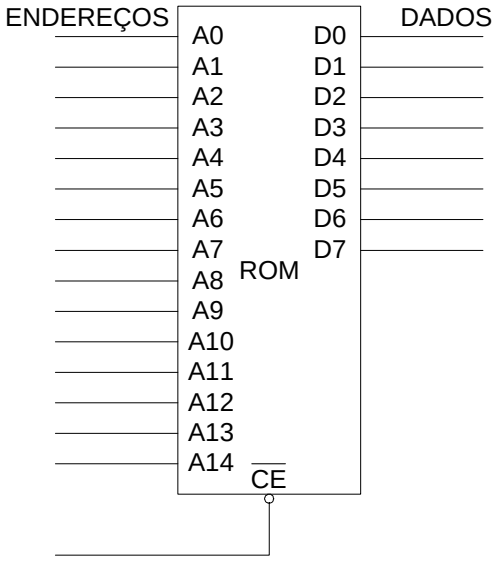


O PC ORIGINAL

TIPOS DE MEMÓRIAS (ROM)

O termo ROM significa Read Only Memory (memória somente de leitura). Existem vários tipos de ROM que podem ser programadas, lidas, apagadas e programadas com novos dados, mas a principal característica da ROM é que não é volátil. Isto significa que os dados podem ser armazenados e não se perdem quando a energia é removida da memória.

A figura mostra o símbolo esquemático de uma ROM. De acordo com o BUS de dados D0 a D7, esta ROM pode armazenar em cada endereço 8 bits de dados. As saídas de dados são TRI-STATE. Isto significa que cada saída pode ter um nível lógico 0, um nível lógico 1, ou uma alta impedância. No estado de alta impedância a saída é desligada do que estiver ligado a ela. Se a entrada $\overline{CE}$ da ROM não estiver accionada, então todas as saídas de dados estarão em alta impedância. Algumas ROMs também ficam em modo de baixo consumo quando a



entrada $\overline{CE}$ não esta accionada. Se a entrada $\overline{CE}$ é accionada, a ROM é activada, e as saídas também serão activadas. Assim as saídas ficarão a um nível lógico normal 0 ou 1.

Todos os dados na ROM ficam armazenados em forma de uma lista numerada. O número que identifica a localização de cada palavra armazenada na lista é chamado endereço. Podemos dizer o número de palavras armazenadas na ROM pelo número de entradas de endereços. O número de palavras é igual a 2N, N é igual ao número de linhas de endereço. A ROM da figura anterior tem 15 linhas de endereço, A0 até A14, assim o número de palavras é 215 ou 32 768. A data sheet refere-se a este dispositivo como 32K x 8 ROM. Isto significa que tem 32K de endereços com 8 bits por endereço.

Para obter uma palavra na saída da ROM, tem que fazer duas coisas. Tem que aplicar o endereço nas entradas de endereços, A0 a A14, e accionar a entrada $\overline{CE}$.

Agora vamos ver para que são necessárias as saídas TRI-STATE numa ROM. Vamos supor que queremos guardar mais que 32K de dados. Nós podemos fazer isto conectando duas ou mais ROMs em paralelo, de modo que podemos endereçar um dos 32 768 endereços em cada uma. O conjunto de linhas paralelas usado para enviar o endereço ou dados é chamado BUS. As saídas de dados das ROMs são ligados em paralelo de modo que qualquer uma das ROMs possa enviar os dados para um BUS de dados comum. Se estas ROMs tiverem apenas as saídas normais com dois estados, irá ocorrer um problema sério quando ambas as ROMs tentarem enviar dados para o BUS. Esta ligação entre as saídas provavelmente irá destruir algumas saídas e fornecer informação sem qualquer significado. Uma vez que as ROMs têm saídas com TRI-STATE, nós podemos usar circuitos externos para assegurar que apenas uma ROM de cada vez seja activada. Há um princípio importante aqui, sempre que houver varias saídas ligadas ao BUS, as saídas devem ser TRI-STATE, e apenas uma ROM deve ser activada de cada vez.

No início deste capítulo foi mencionado que algumas ROMs podem ser apagadas e reprogramadas com novos dados. Em seguida estão descritos alguns tipos de ROMs.

ROM – Programada durante o fabrico.

PROM – Programada pelo utilizador colocando os 1s a 0s. Não pode ser alterada excepto para colocar outros 1s a 0s.

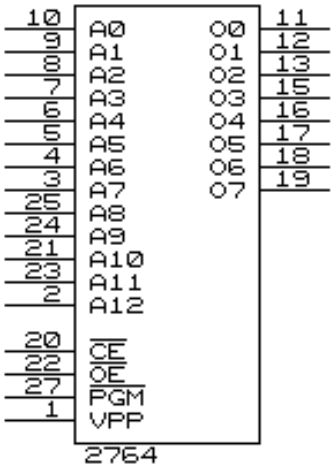
EPROM – Programada electricamente pelo utilizador colocando os 1s a 0s; pode ser apagada com luz ultravioleta através da janela de quartzo.

EEPROM – Programada electricamente pelo utilizador; pode ser apagada com sinais eléctricos de modo a ser reprogramada no circuito.

O $\overline{OE}$ (Output Enable) é colocado a 0 sempre que se deseja ler algo da RAM.

E(PROM) (2764)

Esta memória utiliza 13 linhas de endereço (A0 a A12), permitindo assim o acesso a 8KBytes de endereços, e 8 linhas de dados (D0 a D7).



Existem 4 terminais de controlo ($\overline{CE}, \overline{OE}, \overline{PGM}, VPP$), o $\overline{CE}$ (Chip Enable) deve estar a 0 para a memória ser seleccionada.

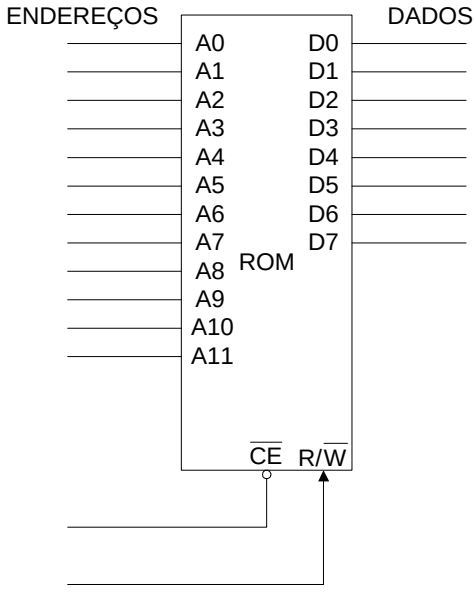
O $\overline{OE}$ (Output Enable) é colocado a 0 sempre que se deseja ler algo da ROM.

No $\overline{PGM}$ (ProGraM) coloca-se um 0 e no VPP coloca-se a tensão de programação sempre que se deseja programar uma nova E(PROM).

TIPOS DE MEMÓRIAS (RAM)

O nome RAM significa Random Access Memory (memória de acesso aleatório), mas uma vez que as ROMs também são de acesso aleatório, o nome deveria ser memória de escrita e leitura. As RAMs são usadas também para armazenar dados em binário. A RAM estática é essencialmente uma matriz de flip-flops. Assim, nós podemos dados num endereço da RAM em qualquer altura aplicando os dados nas entradas de dados e accionando os flip-flops. A informação guardada nos flip-flops mantém-se enquanto a energia estiver aplicada na RAM. Esta memória é volátil porque os dados perdem-se quando a energia é desligada.

A figura seguinte mostra o símbolo esquemático de uma RAM comum. Esta RAM tem 12 linhas de endereço, A0 até A11, por isso pode armazenar 2^{12} (4096) palavras de dados. As oito linhas de dados indicam que a RAM pode armazenar 8 bits em cada endereço. Quando estamos a escrever na RAM, estas linhas funcionam como entradas. A entrada $\overline{CE}$, é usada para activar a RAM para escrita ou leitura. A entrada R/$\overline{W}$ é colocada a lógico 1 para ler da RAM e colocada a lógico 0 para escrever na RAM. Aqui está como todas estas linhas

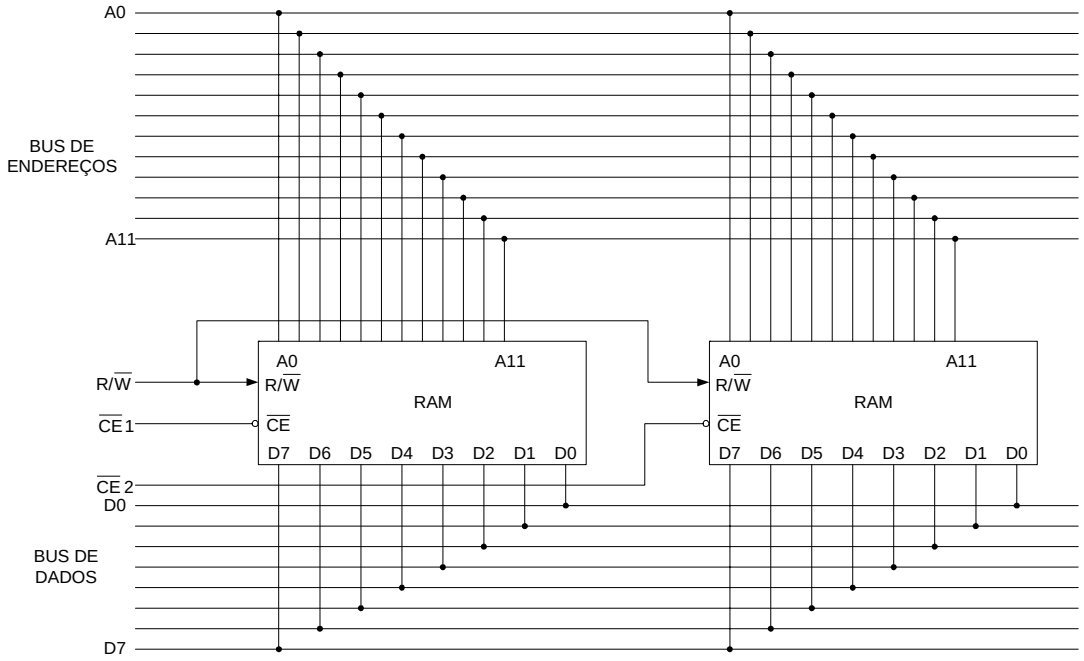


funcionam para ler e escrever neste dispositivo.

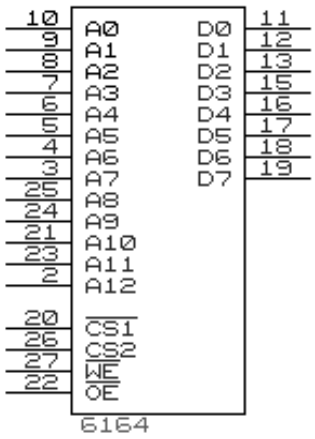
Para escrever na RAM, aplicamos o endereço desejado nas entradas de endereço, colocamos a entrada $\overline{CE}$ a lógico 0 para accionar o dispositivo, e colocamos a entrada $R/\overline{W}$ a lógico 0 para indicar à RAM que queremos escrever nela. Aplicamos então os dados que queremos escrever nas linhas de dados durante um tempo específico. Para ler informação da RAM, indicamos o endereço desejado, colocamos a entrada $\overline{CE}$ a lógico 0, e colocamos a entrada $R/\overline{W}$ a lógico 1 para indicar à RAM que queremos ler. Para uma operação de leitura as saídas de dados estarão activas para fornecer os dados que estão no endereço indicado pelas linhas de endereço.

As memórias estáticas SRAM que acabamos de ver guardam os dados numa matriz de flip-flops. Nas memórias dinâmicas DRAM, os dados binários 0s e 1s são guardados numa carga eléctrica num condensador. Uma vez que estes condensadores ocupam menos espaço que um flip-flop, uma RAM dinâmica pode armazenar muito mais bits que uma memória estática do mesmo tamanho. A desvantagem das memórias dinâmicas é que os condensadores descarregam-se. O estado lógico armazenado em cada condensador deve ser refrescado todos os 2 milisegundos.

A memória RAM pode ser ligada em paralelo do mesmo modo que a ROM. Na figura seguinte está representa a ligação de duas RAMs em paralelo. O funcionamento do BUS é igual ao esquema com ROMs. Neste esquema com RAMs existe a entrada $R/\overline{W}$ que é comum a todas as RAMs que estiverem ligadas em paralelo. Apenas uma RAM pode ser seleccionada através da entrada $\overline{CE}$.



RAM (6164)



Esta memória utiliza 13 linhas de endereço (A0 a A12), permitindo assim o acesso a 8KBytes de endereços, e 8 linhas de dados (D0 a D7).

Existem 4 terminais de controlo ($\overline{CS1}, \overline{CS2}, \overline{WE}, \overline{OE}$), estes 4 terminais são activos a 0, o $\overline{CS1}$ e $\overline{CS2}$ (Chip Select) devem estar ambos a 0 para a memória ser seleccionada.

$\overline{WE}$ (Write Enable) é colocado a 0 sempre que se deseja escrever algo na RAM.

INSTRUÇÕES MONO E MULTI-ENDEREÇO

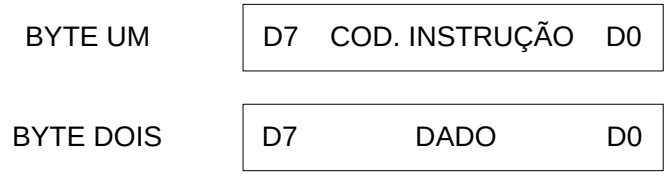
Qualquer processador, por mais sofisticado que seja, só faz aquilo para que for instruído. Por isso, os construtores de um processador projectam-no de acordo com um determinado grupo de instruções, cada uma delas com uma função específica, de modo que, quando uma dessas instruções é introduzida no processador, leva à obtenção de determinados resultados previamente estudados.

O conhecimento da função de cada instrução permitir-nos-á associá-las de modo a construir programas que resolvam os problemas mais variados e complexos.

FORMATO DAS INSTRUÇÕES – O grupo básico de bits que um processador trata (lê, escreve, opera, etc.) de uma só vez, constitui uma unidade que caracteriza esse processador. Assim, se um processador consegue tratar um conjunto de 16 bits em um só passo, diz-se que é um processador de 16 bits. Se o processador consegue tratar um conjunto de 8 bits, diz-se que é um processador de 8 bits.

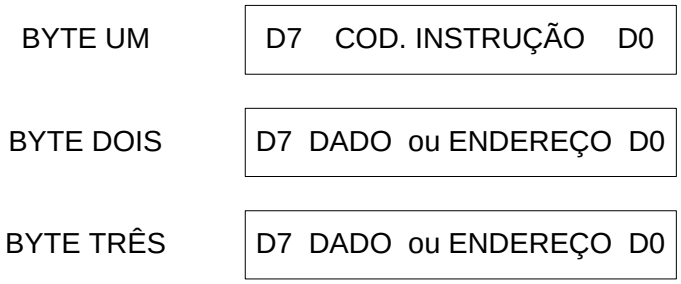
O 8085 é caracterizado por ser de 8 bits. As unidades de memória a ele associadas devem, então, estar organizadas em posições endereçáveis que possam armazenar, cada uma, um byte. Por isso, o formato dos dados e instruções do 8085 apresentam-se de modo que o byte constitui sempre a unidade processável em cada passo do processamento.

As instruções do 8085 podem apresentar um formato de um, dois, ou três bytes. As instruções que contêm mais do que um byte são colocadas na memória de modo que os bytes ocupam posições de memória consecutivas, sendo o endereço do 1º byte da instrução tomado como o endereço da própria instrução. Como se vê na figura seguinte, o primeiro byte de uma instrução, qualquer que seja o formato, é sempre o código da instrução.



INSTRUÇÃO DE 2 BYTES

Nas instruções com 3 bytes, os dois últimos podem ser endereços ou dados. De qualquer modo, a informação contida nesses dois bytes é disposta de modo que os 8 bits menos significativos são colocados na posição imediatamente a seguir ao código de instrução e os 8 bits mais significativos a seguir aos menos significativos.



INSTRUÇÃO DE 3 BYTES

MODOS DE ENDEREÇAMENTO – Um dos processos de classificação das instruções, é feito de acordo com o método de endereçamento dos operandos de que cada instrução necessita para o seu processamento. Façamos, então, um breve estudo dos modos de endereçamento das instruções do 8085.

a) Endereçamento implícito – O modo de endereçamento de certas instruções do 8085 encontra-se implícito na própria função da instrução, não havendo necessidade de qualquer campo de endereçamento, já que a instrução só interfere com um elemento (ou parte) específico do hardware.

Exemplo: STC (Set Carry Flag).

Esta instrução apenas interfere com a Flag Carry, colocando-a a 1.

b) Endereçamento por registo – Uma grande parte das instruções de 8085 utiliza o endereçamento por registo; isto é, a própria instrução especifica, além do código de instrução, um dos registos A, B, C, D, E, H ou L, que conterá um operando no qual o operando vai ser colocado. O acumulador é normalmente utilizado como suporte do segundo operando, quando este é necessário.

Exemplo: CMP Reg (Compare Register)

Esta instrução compara o conteúdo do registo indicado com o conteúdo do acumulador.

c) Endereçamento imediato – Quando uma instrução possui, associado a si, o operando que vai utilizar, diz-se que possui endereçamento imediato.

Exemplo: CPI (Compare Immediate)

Ao executar esta instrução, o 8085 irá comparar o conteúdo do acumulador com o valor expresso no operando associado à instrução.

Numa instrução deste tipo, o 8085 analisa o primeiro byte da instrução (código de instrução) e fica a saber que tem que buscar de imediato o byte seguinte (operando), afim de proceder à execução da instrução.

d) Endereçamento directo – Para ter este modo de endereçamento, a instrução terá que apresentar um formato de três bytes: um para o código da instrução e os outros dois para conterem o endereço (16 bits) que servirá para executar a instrução definida no código de instrução. O modo de endereçamento directo é feito pela própria instrução através dos 2 bytes adicionais que suportam o endereço do operando.

Exemplo: JMP endereço (Jump)

Esta instrução coloca o valor indicado nos dois bytes após o código de instrução no Program Counter.

e) Endereçamento indirecto por registo – Este modo de endereçamento é caracterizado por um processo em que a instrução aponta para uma posição de memória através do endereço colocado num par de registos.

Exemplo: MOV M, Reg

Esta instrução usa o par de registos HL para indicar o endereço de memória no processo de transferencia de dados.

f) Endereçamento misto – Existem algumas instruções do 8085 que utilizam uma combinação dos modos de endereçamento.

A instrução CALL por exemplo utiliza uma combinação do endereçamento directo com o endereçamento indirecto por registo. O modo de endereçamento directo serve para especificar a localização da 1ª instrução da subrotina a ser chamada; o modo de endereçamento indirecto por registo é feito através do Stack (pilha) que serve para guardar o Program Counter para permitir o regresso ao programa principal com a instrução RET.

GRUPO DE TRANSFERENCIA DE DADOS

MOV Reg1, Reg2 (Move Register)

Reg2 $\rightarrow$ Reg1

O conteúdo do Reg2 é copiado para o Reg1.

Ciclos: 1

Endereçamento: de registo

Flags: nenhuma

MOV Reg, M (Move from Memory)

(HL) → Reg

O conteúdo da posição de memória indicada em HL é copiado para Reg.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: nenhuma

MOV M, Reg (Move to memory)

Reg → (HL)

O conteúdo de Reg é copiado para a posição de memória indicada em HL.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: nenhuma

MVI Reg, dado (Move immediate)

Dado → Reg

O dado (2º byte) é colocado em Reg.

Ciclos: 2

Endereçamento: imediato

Flags: nenhuma

MVI M, dado (Move to memory immediate)

dado → (HL)

O dado (2º byte) é colocado na posição de memória indicada em HL.

Ciclos: 3

Endereçamento: imediato/indirecto por registo

Flags: nenhuma

LXI Reg. Pair, word (Load register pair immediate)

Byte3 → Reg. high

Byte2 → Reg. low

O conteúdo do 3º byte é colocado no registo mais significativo e o 2º byte é colocado no registo menos significativo do par de registos.

Ciclos: 3

Endereçamento: imediato

Flags: nenhuma

LDA ender (Load accumulator direct)

(byte3 byte2) → A

O conteúdo da posição de memória indicada por byte 3 e byte2 é copiada para o acumulador.

Ciclos: 4

Endereçamento: directo

Flags: nenhuma

STA ender (Store accumulator direct)

A → (byte3 byte2)

O conteúdo do acumulador é copiado para a posição de memória indicada por byte 3 e byte2.

Ciclos: 4

Endereçamento: directo

Flags: nenhuma

LHLD ender (Load H and L direct)

(byte3 byte2) → L

(byte3 byte2+1) → H

O conteúdo da posição de memória indicada por byte 3 e byte2 é copiado para L, o conteúdo da posição de memória indicada por byte 3 e byte2 +1 é copiado para H.

Ciclos: 5

Endereçamento: directo

Flags: nenhuma

LDAX Reg. Pair (Load accumulator indirect)

(Reg. Pair) → A

O conteúdo da posição de memória, cujo endereço se encontra especificado no par de registos Reg. Pair, é transferido para o acumulador, Reg. Pair apenas se refere aos pares BC e DE.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: nenhuma

STAX Reg.pair (Store accumulator indirect)

A $\rightarrow$ (Reg. Pair

O conteúdo do acumulador é transferido para a posição de memória cujo endereço se encontra no par de registos Reg. Pair. Apenas os pares BC e DE podem usados.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: nenhuma

XCHG (Exchange H and L with D and E)

H $\rightarrow$ D

L $\rightarrow$ E

O conteúdo do par de registos HL é trocado com o conteúdo do par de registos DE.

Ciclos: 1

Endereçamento: de registo

Flags: nenhuma

GRUPO ARITMÉTICO

Este grupo de instruções realiza operações aritméticas com dados contidos nos registos ou na memória.

As instruções deste grupo afectam, regra geral, todas as flags de acordo com o resultado da operação efectuada e as regras relativas aos indicadores de condição que expusemos anteriormente. Existem, no entanto, algumas instruções deste grupo que não afectam todas as flags, este facto será de fácil verificação através da alínea flags após o formato de cada instrução.

As operações de subtracção são realizadas sempre pela utilização do complemento para 2. Verifica-se também que quando existe pedido de empréstimo a flag carry é colocada a 1, caso contrário é colocada a 0.

ADD reg. (Add register)

A + reg. $\rightarrow$ A

O conteúdo do registo reg. É adicionado ao conteúdo do acumulador. O resultado desta operação é guardado no acumulador.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

ADD M (Add memory)

A + (H L) $\rightarrow$ A

O conteúdo da posição de memória, cujo endereço é dado pelo conteúdo do par HL, é adicionado ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

ADI dado (Add immediate)

A + dado $\rightarrow$ A

O conteúdo do segundo byte da instrução é adicionado ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: Imediato

Flags: Z, S, P, C, Ac

ADC reg. (Add register with carry)

A + reg. + Carry $\rightarrow$ A

O conteúdo do registo e o valor da flag C são adicionados ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

ADC M (Add memory with carry)

A + (H L) + Carry $\rightarrow$ A

O conteúdo da posição de memória, cujo endereço se encontra no par de registos HL, é adicionado com o valor da flag C ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

ACI dado (Add immediate with carry)

A + dado + carry$\rightarrow$ A

O conteúdo do segundo byte da instrução é adicionado com o valor da flag C ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: imediato

Flags: Z, S, P, C, Ac

SUB reg. (Subtract register)

A – reg. → A

O conteúdo do registo é subtraído ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

SUB M (Subtract memory)

A - (H L) → A

O conteúdo da posição de memória, cujo endereço é dado pelo conteúdo do par HL, é subtraído ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

SUI dado (Subtract immediate)

A - dado → A

O conteúdo do segundo byte da instrução é subtraído ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: Imediato

Flags: Z, S, P, C, Ac

SBB reg. (Subtract register with borrow)

A - reg. - Carry $\rightarrow$ A

O conteúdo do registo e o valor da flag C são subtraídos ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

SBB M (Subtract memory with borrow)

A - (H L) - Carry $\rightarrow$ A

O conteúdo da posição de memória, cujo endereço se encontra no par de registos HL, é subtraído com o valor da flag C ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

SBI dado (Subtract immediate with borrow)

A - dado – carry $\rightarrow$ A

O conteúdo do segundo byte da instrução é subtraído com o valor da flag C ao conteúdo do acumulador. O resultado desta operação é colocado no acumulador.

Ciclos: 2

Endereçamento: imediato

Flags: Z, S, P, C, Ac

INR reg. (Increment register)

reg. + 1 → reg.

O conteúdo do registo é incrementado de uma unidade. Todas a flags são afectadas, excepto a flag C.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, Ac

INR M (Increment memory)

(HL) + 1 → (HL)

O conteúdo da posição de memória, cujo endereço se encontra no par de registos HL, é incrementado de uma unidade. Todas a flags são afectadas, excepto a flag C.

Ciclos: 3

Endereçamento: indirecto por registo

Flags: Z, S, P, Ac

DCR reg. (Decrement register)

reg. - 1 → reg.

O conteúdo do registo é decrementado de uma unidade. Todas a flags são afectadas, excepto a flag C.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, Ac

DCR M (Decrement memory)

(HL) - 1 → (HL)

O conteúdo da posição de memória, cujo endereço se encontra no par de registos HL, é decrementado de uma unidade. Todas a flags são afectadas, excepto a flag C.

Ciclos: 3

Endereçamento: indirecto por registo

Flags: Z, S, P, Ac

INX reg. pair (Increment register pair)

reg. hi reg. low + 1 → reg. hi reg. low

O conteúdo do par de registos é incrementado de uma unidade. Nenhuma flag é afectada.

Ciclos: 1

Endereçamento: de registo

Flags: nenhuma

DCX reg. pair (Decrement register pair)

reg. hi reg. low - 1 → reg. hi reg. low

O conteúdo do par de registos é decrementado de uma unidade. Nenhuma flag é afectada.

Ciclos: 1

Endereçamento: de registo

Flags: nenhuma

DAD reg. pair (Add register pair to H and L)

reg. hi reg. low + HL → HL

O conteúdo do par de registos indicado é adicionado ao conteúdo do par de registos HL. O resultado desta operação é colocado no par de registos HL. Apenas a flag C é afectada por esta instrução.

Ciclos: 3

Endereçamento: de registo

Flags: C

DAA (Decimal adjust Accumulator)

O número de oito bits do acumulador é ajustado de modo a formar dois digitos BCD de quatro bits, através do processo seguinte:

Se o valor expresso pelos quatro bits menos significativos do acumulador for maior do que nove ou se a flag Ac estiver a 1, então adiciona-se 6 ao acumulador.

Se o valor dos quatro bits mais significativos do acumulador for, agora, maior do que nove ou se a flag C=1, então adiciona-se 6 ao valor dos 4 bits mais significativos do acumulador.

Ciclos: 1

Endereçamento: implícito

Flags: Z, S, P, C, Ac

CMP reg. (Compare register)

A – reg.

O conteúdo do registo é subtraído ao conteúdo do acumulador. O conteúdo do acumulador permanece inalterado.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

CMP M (Compare memory)

A - (H L)

O conteúdo da posição de memória, cujo endereço é dado pelo conteúdo do par HL, é subtraído ao conteúdo do acumulador. O conteúdo do acumulador permanece inalterado.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

CPI dado (Compare immediate)

A – dado

O conteúdo do segundo byte da instrução é subtraído ao conteúdo do acumulador. O conteúdo do acumulador permanece inalterado.

Ciclos: 2

Endereçamento: Imediato

Flags: Z, S, P, C, Ac

GRUPO LÓGICO

Este grupo de instruções tem como finalidade a realização das operações lógicas sobre os dados que se

encontram nos registos e na memória e ainda sobre o valor do registo de flags.

ANA reg. (AND register)

A AND reg. → A

Esta instrução faz uma operação lógica AND entre o conteúdo do registo e o acumulador. O AND lógico é feito bit a bit e o resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 1.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

ANA M (AND memory)

A AND (HL) → A

Esta instrução faz uma operação lógica AND entre o conteúdo do acumulador e o conteúdo da posição de memória cujo endereço se encontra no par HL. O resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 1.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

ANI dado (AND dado)

A AND dado → A

Esta instrução faz uma operação lógica AND entre o conteúdo do registo e o valor do segundo byte da instrução. O resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 1.

Ciclos: 2

Endereçamento: imediato

Flags: Z, S, P, C, Ac

XRA reg. (Exclusive OR register)

A XOR reg. → A

Esta instrução faz uma operação lógica XOR entre o conteúdo do registo e o acumulador. O XOR lógico é feito bit a bit e o resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 0.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

XRA M (Exclusive OR memory)

A XOR (HL) → A

Esta instrução faz uma operação lógica XOR entre o conteúdo do acumulador e o conteúdo da posição de memória cujo endereço se encontra no par HL. O resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 0.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

XRI dado (XOR dado)

A XOR dado → A

Esta instrução faz uma operação lógica XOR entre o conteúdo do registo e o valor do segundo byte da instrução. O resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 0.

Ciclos: 2

Endereçamento: imediato

Flags: Z, S, P, C, Ac

ORA reg. (OR register)

A OR reg. → A

Esta instrução faz uma operação lógica OR entre o conteúdo do registo e o acumulador. O OR lógico é feito bit a bit e o resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 0.

Ciclos: 1

Endereçamento: de registo

Flags: Z, S, P, C, Ac

ORA M (OR memory)

A OR (HL) → A

Esta instrução faz uma operação lógica OR entre o conteúdo do acumulador e o conteúdo da posição de memória cujo endereço se encontra no par HL. O resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 0.

Ciclos: 2

Endereçamento: indirecto por registo

Flags: Z, S, P, C, Ac

ORI dado (OR dado)

A XOR dado → A

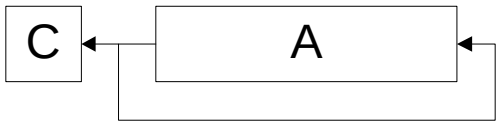
Esta instrução faz uma operação lógica XOR entre o conteúdo do registo e o valor do segundo byte da instrução. O resultado é colocado no acumulador. Após a execução desta instrução a flag C encontra-se a 0 e a flag Ac a 0.

Ciclos: 2

Endereçamento: imediato

Flags: Z, S, P, C, Ac

RLC (Rotate left)



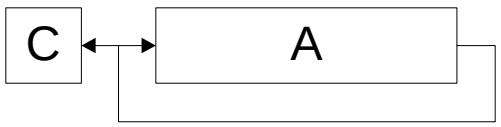
O conteúdo do acumulador é rodado uma posição (bit) para a esquerda. O conteúdo do bit mais significativo é colocado no bit menos significativo e na flag C. Somente a flag C é afectada por esta instrução.

Ciclos: 1

Endereçamento: implícito

Flags: C

RRC (Rotate right)



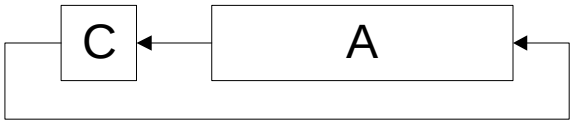
O conteúdo do acumulador é rodado uma posição (bit) para a direita. O conteúdo do bit menos significativo é colocado no bit mais significativo e na flag C. Somente a flag C é afectada por esta instrução.

Ciclos: 1

Endereçamento: implícito

Flags: C

RAL (Rotate left through carry)



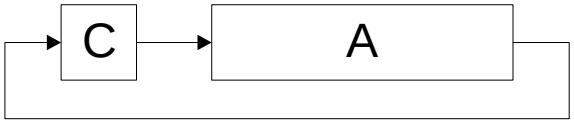
O conteúdo do acumulador é rodado uma posição (bit) para a esquerda através da flag carry. O conteúdo do bit mais significativo é colocado na flag carry e a flag carry é colocada no bit menos significativo. Somente a flag C é afectada por esta instrução.

Ciclos: 1

Endereçamento: implícito

Flags: C

RAR (Rotate right through carry)



O conteúdo do acumulador é rodado uma posição (bit) para a direita através da flag carry. O conteúdo do bit menos significativo é colocado na flag carry e a flag carry é colocada no bit mais significativo. Somente a flag C é afectada por esta instrução.

Ciclos: 1

Endereçamento: implícito

Flags: C

CMA (Complement accumulator)

$\overline{A} \rightarrow A$

O conteúdo do acumulador é complementado (onde existem zeros passa a existir uns e vice-versa). Nenhuma flag é afectada.

Ciclos: 1

Endereçamento: implícito

Flags: Nenhuma

CMC (Complement carry)

$\overline{C} \rightarrow C$

A flag C é complementada. Nenhuma outra flag é afectada.

Ciclos: 1

Endereçamento: implícito

Flags: C

STC (Set carry)

$1 \rightarrow C$

A flag C é colocada a 1. Nenhuma outra flag é afectada.

Ciclos: 1

Endereçamento: implícito

Flags: C

GRUPO DE TRANSFERÊNCIA DE CONTROLO

Este grupo de instruções altera a sequência normal do programa. Não há nenhuma instrução deste grupo que afecte qualquer das flags.

Existem dois tipos de instruções neste grupo: instruções de transferência condicional e instruções de transferência incondicional. As instruções de transferência incondicional limitam-se a alterar o conteúdo do Program Counter e colocar neste o endereço especificado nos dois bytes que seguem o código de instrução. As instruções de transferência condicional fazem a análise do estado de uma flag e mediante esse estado tomam a decisão de transferir ou não para o Program Counter o endereço especificado nos dois bytes que seguem o código de instrução.

As condições que determinam a transferência de controlo encontram-se especificadas na própria instrução:

NZ – not zero (Z = 0)

Z – zero (Z = 1)

NC – no carry (C = 0)

C – carry (C = 1)

PO – parity odd (P = 0)

PE – parity even (P = 1)

P – plus (S = 0)

M – minus (S = 1)

JMP ender (Jump)

ender → PC

O valor do endereço é colocado no Program Counter incondicionalmente.

Ciclos: 1

Endereçamento: imediato

Flags: Nenhuma

JNZ, JZ, JNC, JC, JPO, JPE, JP, JM ENDER (Conditional jump)

Se a condição da instrução é valida então ender → PC, caso contrário a sequência do programa segue para a instrução seguinte.

Ciclos: 2/3

Endereçamento: imediato

Flags: Nenhuma

CALL ender (Call)

PC high → (SP – 1)

PC low → (SP – 2)

SP - 2 → SP

ender → PC

Os oito bits mais significativos do endereço da instrução seguinte é transferido para a posição de memória cujo endereço é determinado pelo conteúdo do ponteiro de pilha (SP) menos uma unidade. Os oito bits menos significativos do endereço daquela instrução é colocado na posição de memória cujo endereço é definido pelo conteúdo do ponteiro (SP) de pilha menos 2 unidades. O valor do endereço é colocado no Program Counter.

Ciclos: 5

Endereçamento: imediato, indirecto e por registo

Flags: Nenhuma

CNZ, CZ, CNC, CC, CPO, CPE, CP, CM ENDER (Conditional Call)

Se a condição da instrução é valida então é executada a operação CALL, caso contrário a sequência do programa segue para a instrução seguinte.

Ciclos: 5

Endereçamento: imediato

Flags: Nenhuma

RET (Return)

(SP) → PCL

(SP+1) → PCH

SP=SP+2

O conteúdo da posição de memória cujo endereço está contido no ponteiro de pilha, é transferido para o Program Counter. O conteúdo do ponteiro da pilha é incrementado de 2 unidades.

RNZ, RZ, RNC, RC, RPO, RPE, RP, RM ENDER (Conditional Return)

Se a condição da instrução é valida então é executada a operação Return, caso contrário a sequência do programa segue para a instrução seguinte.

Ciclos: 3

Endereçamento: imediato

Flags: Nenhuma

RST n (Restart)

PC high → (SP – 1)

PC low → (SP – 2)

SP - 2 → SP

8 x n → PC

O Program Counter é colocado na pilha e o ponteiro de pilha é decrementado 2 unidades. O Program Counter recebe um valor que é igual a oito vezes o número representado por n.

Ciclos: 3

Endereçamento: indirecto por registo

Flags: Nenhuma

PCHL (Jump H and L indirect move H and L to PC)

(L) → PCL

(H) → PCH

O conteúdo do registo H é transferido para os oito bits mais significativos do Program Counter. O conteúdo do registo L é transferido para os oito bits menos significativos do Program Counter.

Ciclos: 1

Endereçamento: de registo

Flags: Nenhuma

GRUPO DE CONTROLO DE ENTRADA E SAÍDA

Este grupo de instruções executa operações de entrada e saída, manipula a pilha (stack) e altera os indicadores internos de controlo.

PUSH reg. Pair

reg. H → (SP)

reg. L → (SP - 1)

SP=SP – 2

O conteúdo do registo mais significativo do par de registos é transferido para a posição de memória com endereço igual ao conteúdo do ponteiro de pilha subtraído de uma unidade. O conteúdo do registo menos significativo do par de registos é transferido para a posição de memória com o endereço igual ao conteúdo do ponteiro de pilha subtraído de 2 unidades. O ponteiro de pilha é decrementado 2 unidades.

Ciclos: 3

Endereçamento: indirecto por registo

Flags: Nenhuma

PUSH PSW (Processor Status Word)

A → (SP)

Flags → (SP+1)

SP=SP - 2

O conteúdo do acumulador é transferido para a posição de memória com endereço igual ao conteúdo do ponteiro de pilha subtraído de uma unidade. O conteúdo do registo de flags é transferido para a posição de memória com o endereço igual ao conteúdo do ponteiro de pilha subtraído de 2 unidades. O ponteiro de pilha é decrementado 2 unidades.

Ciclos: 3

Endereçamento: indirecto por registo

Flags: Nenhuma

POP reg. Pair

(SP) → reg. H

(SP+1) → reg. L

SP=SP + 2

O conteúdo da posição de memória, cujo endereço se encontra no ponteiro de pilha, é transferido para o registo menos significativo do par de registos. O conteúdo da posição de memória, cujo endereço é igual ao

conteúdo do ponteiro de pilha acrescido de uma unidade, é transferido para o registo mais significativo do par de registos. O ponteiro da pilha é incrementado de duas unidades.

Ciclos: 3

Endereçamento: indirecto por registo

Flags: Nenhuma

POP PSW (Processor Status Word)

(SP) $\rightarrow$ A

(SP + 1) $\rightarrow$ Flags

SP=SP + 2

O conteúdo da posição de memória endereçada pelo ponteiro de pilha é transferido para a palavra de estado, afim de recolocar nas flags os valores existiam antes da instrução PUSH PSW. O conteúdo da posição de memória, com endereço igual ao conteúdo do ponteiro de pilha acrescido de uma unidade, é transferido para o acumulador. O ponteiro de pilha é incrementado de duas unidades.

Ciclos: 3

Endereçamento: indirecto por registo

Flags: Nenhuma

XTHL (Exchange stack top with H and L)

(L) $\leftrightarrow$ (SP)

(H) $\leftrightarrow$ (SP+1)

O conteúdo do registo L é trocado com o conteúdo da posição de memória endereçada pelo ponteiro de pilha. O conteúdo do registo H é trocado com o conteúdo da posição de memória com endereço igual ao ponteiro de pilha acrescido de uma unidade

Ciclos: 5

Endereçamento: indirecto por registo

Flags: Nenhuma

SPHL (Move HL to SP)

HL → SP

O conteúdo do par de registos é transferido para o ponteiro de pilha.

Ciclos: 3

Endereçamento: directo

Flags: Nenhuma

IN endereço (Input)

Port (endereço) → A

O dado colocado no bus de dados pelo port do endereço especificado, é transferido para o acumulador.

Ciclos: 3

Endereçamento: directo

Flags: Nenhuma

OUT endereço (Output)

A → Port (endereço)

O conteúdo do acumulador é colocado no bus de dados e enviado para o port de endereço especificado.

Ciclos: 3

Endereçamento: directo

Flags: Nenhuma

TECNOLOGIA USADA EM COMPUTADORES

O PC rapidamente precisou de vários melhoramentos, para acompanhar o rápido desenvolvimento do software, e por isso precisou de mais capacidade na RAM e no disco. Esta necessidade foi ultrapassada com a introdução do PC XT com os seguintes melhoramentos:

Foi aumentado o numero de slots para 8

Foram acrescentados um disco e um adaptador

Foram juntos dois port's, um paralelo e um série

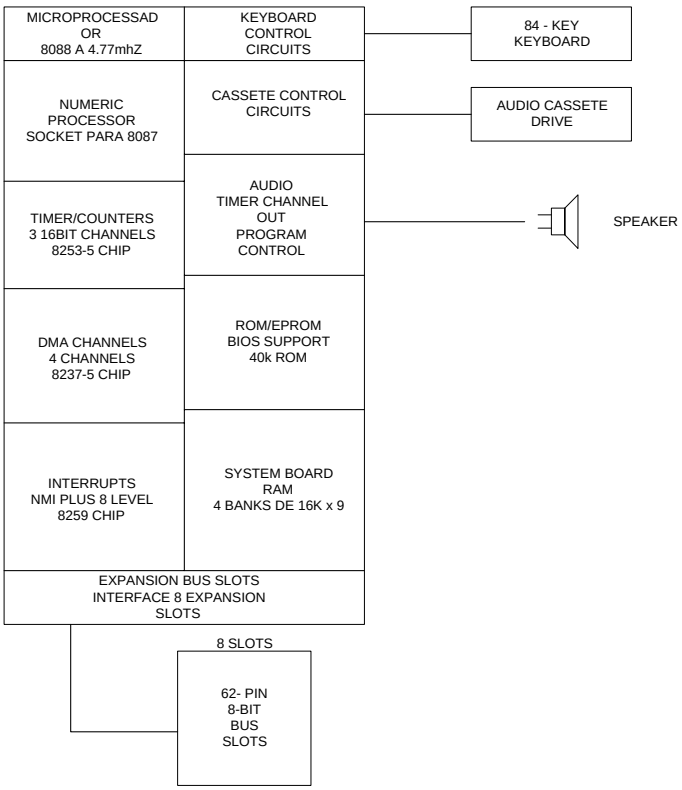
A memória RAM foi aumentada para 256K

A potência da fonte de alimentação foi aumentada de 65 para 135W

A capacidade das disquetes foi aumentada para 360K

O tamanho físico da caixa foi mantido igual ao IBM PC.

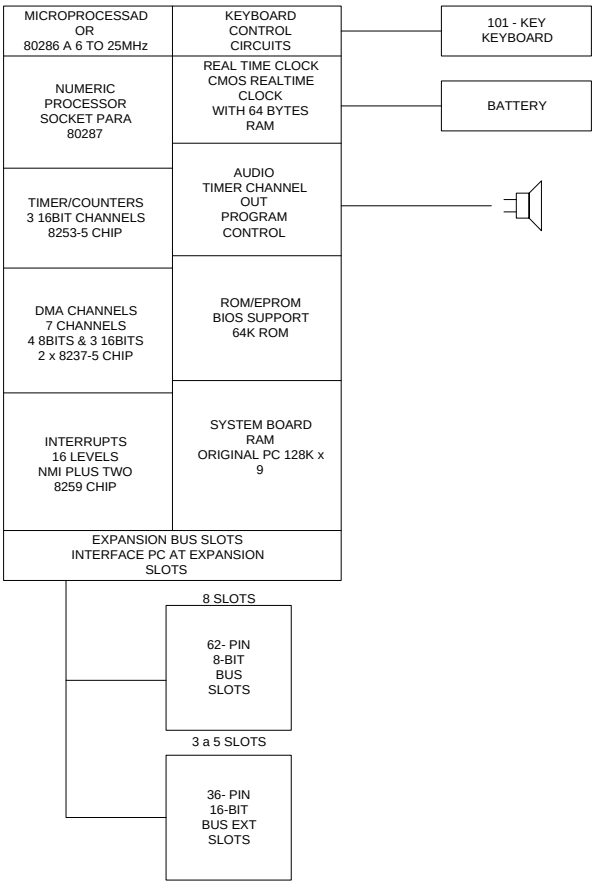
O PC XT não aumentou a velocidade do microprocessador, entretanto outros fabricantes aumentaram a performance do sistema, aumentando a velocidade do clock para 8 e até mesmo 10MHz. Alguns fabricantes mudaram mesmo o microprocessador para o 8086. Uma memória de 16 bits versus uma de 8 bits, resulta em um aumento de performance.



O PC XT

A introdução do PC AT marcou o uso da nova geração de microprocessadores da Intel 80286. Para acomodar este microprocessador de 16 bits de dados e 24 bits de endereços, o AT estendeu o BUS do sistema. Era necessário entretanto manter a compatibilidade com o sistema anterior, por isso as slots de expansão do AT continham as anteriores do PC XT.

80286 ofereceu uma melhor performance, com uma maior frequência de clock, um menor número de clocks por instrução, novas instruções, e um BUS de 16bits. O 80286 suporta dois modos de operação: modo real e protegido. O modo real emula o 8088/8086 com um ambiente de 1Mbyte de RAM, o que permite que o software do 8088/8086 seja executa sem necessitar de alterações. No modo protegido, pode trabalhar num ambiente de até 16Mbytes de RAM e em ambiente multitarefa.

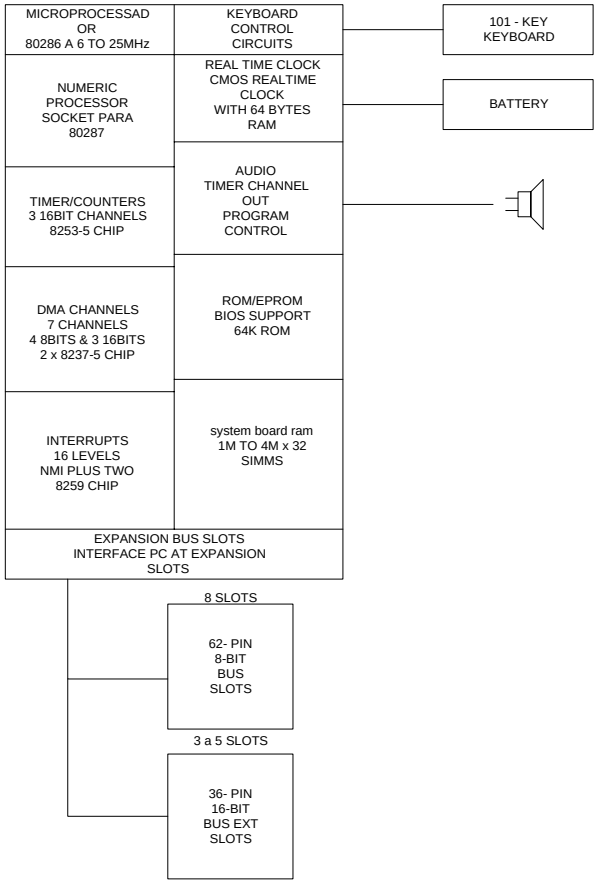


O PC AT

Com a introdução do 286 e do BUS AT, foi introduzido o floppy de 1.2M (disquete flexível). O AT juntou também um relógio com calendário alimentado por uma pilha, e o sistema de vídeo EGA (Enhanced Graphics Adapter).

O passo seguinte foi a introdução do microprocessador de 32 bits da Intel 80386 com a respectiva evolução até aos processadores actuais. A IBM introduziu o 80386 na família PS/2, o resto da indústria manteve o BUS AT a trabalhar com o 80386.

O sistema com o 80386, tem um BUS de dados e de endereços de 32bits.



386 PC AT

ARQUITECTURA E INTERFACES

O 8085 e muitos outros microprocessadores devem começar a executar o programa desde a primeira posição de memória, que é o caso do 8085 que inicia no endereço 0000H. Isto é efectuado por um circuito que aplica um sinal ao terminal do 8085 chamado RESET-IN quando o computador é ligado. Quando o botão PB1 é pressionado ou quando se liga a alimentação, a entrada RESET-IN do 8085 recebe um nível lógico 0. O condensador mantém o Zero durante algum tempo. O valor lógico 0 no terminal RESET-IN causa o reset de algumas das suas partes constituintes e faz com que o Program Counter aponte para o início da memória.

Quando o condensador carrega através de R1, a linha RESET-IN sobe até ao valor lógico 1 desactivando o sinal. O díodo D1 serve para descarregar rapidamente o condensador C1 através da fonte de alimentação unicamente quando a alimentação está desligada. Com RESET-IN desactivado, o 8085 começa a funcionar.

O circuito gerador de clock, cuja frequência é colocada a 6.144MHz pelo cristal Y1, conduz todas as funções de temporização do 8085 e funciona desde que a alimentação esteja aplicada ao CPU. É possível operar o 8085 através de uma gama larga de frequências, mas neste caso o 8085 utiliza 6.144MHz para originar compatibilidade com os circuitos usados no sistema. O sinal do oscilador é dividido por 2 dentro do 8085, e é o clock principal do sistema, referido como SYSCLK. Toda a temporização nas operações realizadas pelo 8085 são conduzidas pelo SYSCLK, que no caso é de 3.072MHz. Mesmo os sinais assíncronos de entrada como o RESET-IN e interrupções são internamente sincronizadas por este oscilador.

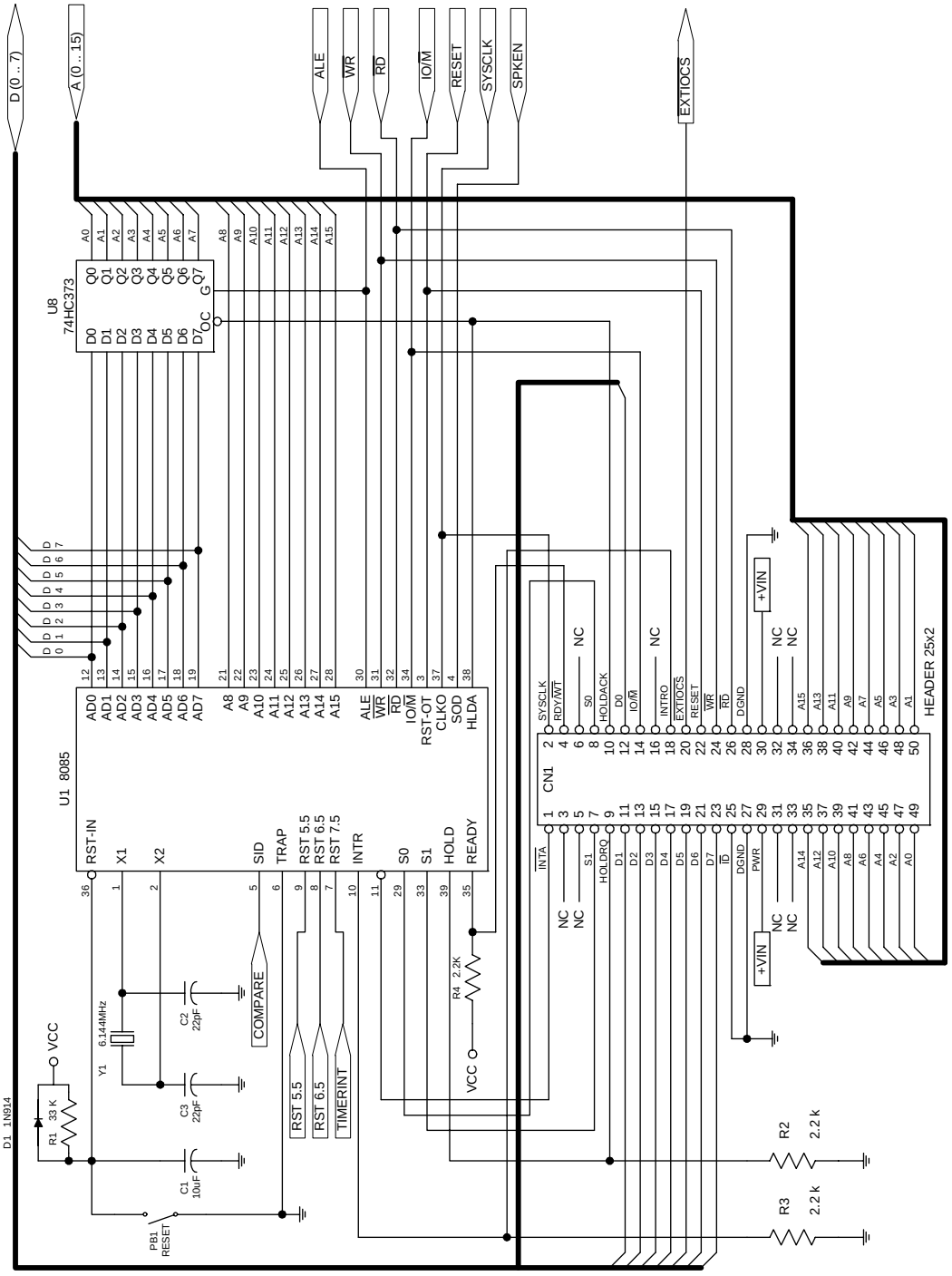
A primeira coisa que o 8085 fará após RESET-IN ser desactivado, é ir buscar uma instrução ao endereço de memória 0000H. Antes da instrução ser lida da memória, o 8085 comuta internamente o BUS de endereços (A0-A7) para BUS de dados (D0-D7), de modo a reduzir o numero de pinos do integrado do 8085, numa operação chamada multiplexagem. A multiplexagem permite que os pinos AD0-AD7 possam ser usados como BUS de dados e também como os 8 bits menos significativos dos 16 bits de endereço. Estes pinos devem ser desmultiplexados antes de se poderem usar adequadamente. O 8085 faz sair o Byte de menor peso do endereço através das linhas AD0-AD7, e o Byte de maior peso directamente através das linhas A8-A15, um sinal chamado Address Latch Enable (ALE) vai a 1 e a 0, originando que o Latch (74HC373) associa-se com as linhas de endereço A8-A15, para dar origem a 16 bits de linhas de endereço para as memórias e descodificadores.

As 16 linhas de endereço são aplicadas ao bloco de memória para seleccionar o endereço que o 8085 deseja ler ou escrever. No caso de ser necessário ler, o sinal $\overline{RD}$ do 8085 será activado. Assim o 8085 poderá ler um Byte da memória. Este Byte vai para o descodificador de instruções, que determinará que instrução é, e seguidamente o 8085 executa-a. Dependendo do tipo de instrução, o 8085 poderá ir buscar dados adicionais, ler/escrever dados a outra posição de memória ou realizar operações de I/O (etc). De cada vez que há transferência de informação no BUS ocorrerão estas operações: o byte menos significativo do endereço será enviado para AD0-AD7, que será memorizado pelo Latch de endereços. Um ciclo completo de leitura, descodificação e execução de uma instrução, forma um “Bus cycle” usando sempre de 3 a 14 impulsos de clock (SYSCLK) dependendo do tipo de instrução.

O 8085 tem 5 terminais dedicados a “interromper” o microprocessador. Estes terminais são chamados: TRAP, RST 7.5, RST 6.5, RST 5.5 e INTR. Quando um sinal é aplicado a um destes terminais uma interrupção é gerada. Uma interrupção (interrupt) é uma função especial do 8085, suspendendo a execução do programa e executando o programa que deu origem à interrupção. Logo que a interrupção termine, o 8085 deve restituir os dados e o estado do programa original, e regressar a este no ponto em que o abandonou. Algumas vezes as interrupções são ignoradas deliberadamente se estiverem mascaradas “Masking”, assim as interrupções podem ser ligadas e desligadas quando necessário, poderá parte de um programa ser tão importante que necessite de não ser interrompido até terminar.

Uma interrupção pode ser originada para atender o telefone enquanto se lê um livro. Quando o telefone

toca, coloca-se o livro na mesa com a página marcada, enquanto se atende o telefone. Quando se acaba a acção anterior, vai-se buscar o livro e continua-se a ler no ponto onde o deixámos. Você pode estar tão envolvido na leitura, que pode ignorar a campainha do telefone, pode “mascarála” (mask). Mas se a ignora por muito tempo, pode perdê-la. Algumas vezes as interrupções podem perder-se também. Algumas interrupções podem ser feitas para durar, normalmente por hardware. Estas interrupções persistem, da mesma maneira que a pessoa que lhe telefonou não desiste enquanto não a atender.



Descodificador I/O e memória.

A secção de descodificação de um sistema com microprocessador faz sair os sinais do Bus de endereços do 8085 e selecciona os circuitos correctos da secção da memória ou de I/O. Grande parte dos descodificadores são um conjunto de portas, usando vários níveis de integração. Como todos os sistemas têm uma diferente configuração de memória e I/O, cada descodificador tem um desenho apropriado. No esquema da Fig. 2, dois tipos de mapas de memória necessitam ser descodificados. No mapa standard, 16Kbytes (1Kbyte=1024 bytes) são reservados para o programa da Eprom, começando no endereço 0000H e finalizando em 3FFFH. Este esquema pode suportar 8K, 16K ou 32K de EPROM. O programa da EPROM na sua versão melhorada é actualmente muito pequeno, necessitando menos que 4K, mas os 8K de EPROM são o menor circuito compatível que pode ser utilizado.

Quase todos os programas podem necessitar de memória que pode ser lida como também escrita.

O socket para 32Kbytes de memória permite a utilização de uma RAM estática. Os jumpers de configuração, OJ2 e OJ3 resolvem as diferenças de hardware entre 8K, 16 e 32K de EPROM e permite a adição de 32K Bytes de memória através do arranjo do mapa de memória para acomodar os diferentes circuitos. A memória pode ser alterada para usar os dois mapas de memória descritos. No caso da EPROM, 8K (2764) ou de 16K (27128) podem ser usados alternadamente, tendo em consideração que a de menor tamanho não preenche o mapa inteiro. Quando um mapa alternativo é usado, uma memória de 32K (27256) deve ser instalada no socket da EPROM. A diferença dos terminais de 32K é a razão dos jumpers de opção. O jumper OJ2 liga o terminal 27 entre A14 ou VCC, como necessário pelo dispositivo de 32K.

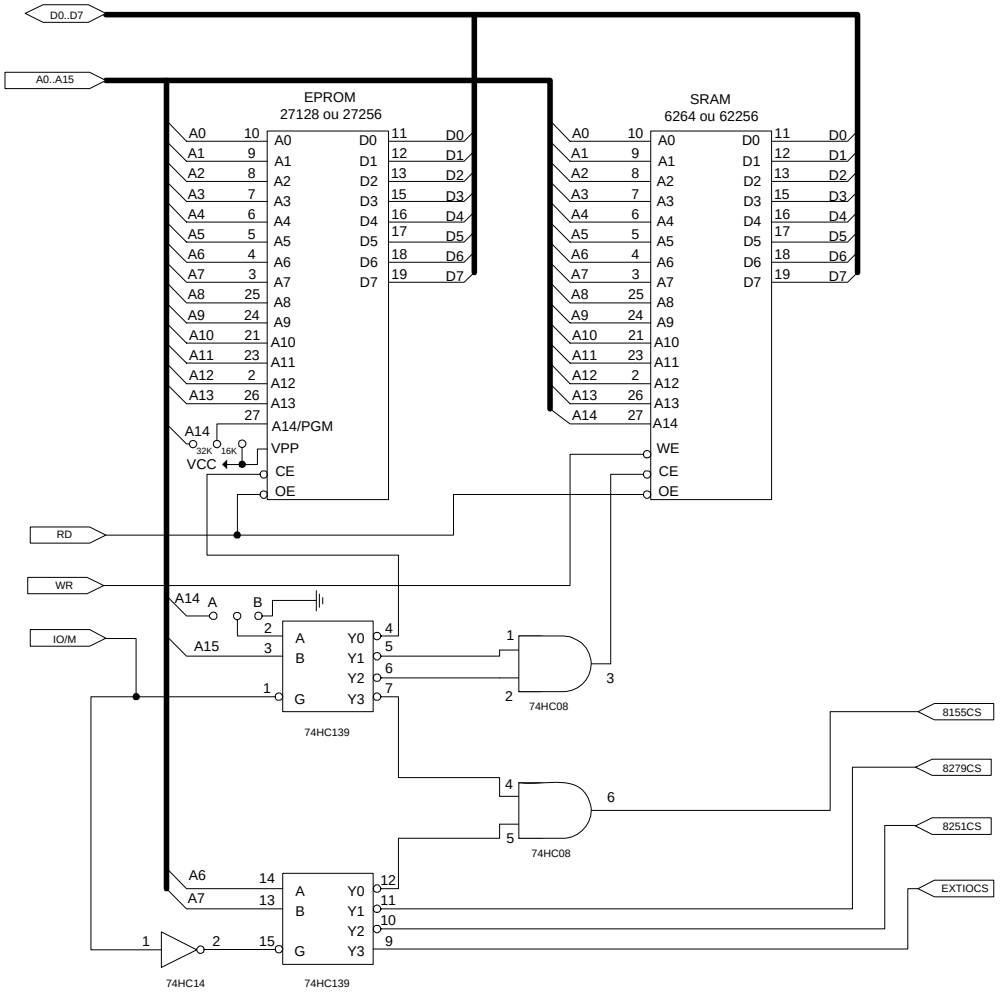
Dois descodificadores (74139), descodificam as linhas adequadas de endereço do 8085 para seleccionar os circuitos requeridos. O descodificador de memória observa o código das linhas de endereço A14 e A15. O estado destas linhas representam intervalos de 16K de endereço. Para modificar o intervalo para 32K Bytes, o terminal do descodificador que está ligado a A14 deve em vez disso ligar à massa. O descodificador I/O observa o estado das linhas de endereço A6 e A7.

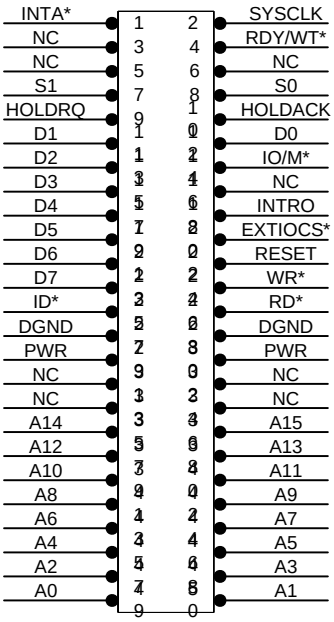
O mapa de I/O tem apenas um comprimento de 256 endereços. O descodificador I/O está a ser utilizado na sua máxima capacidade seleccionando 64 endereços por saída.

A linha de saída do 8085 $IO/\overline{M}$ diz aos descodificadores que é um endereço de I/O se está a 1, ou que é um endereço de memória, se está a 0. Um simples inversor troca a polaridade de $IO/\overline{M}$ para $\overline{IO}/M$. Assim cada descodificador tem o sinal correcto. Todas as saídas do descodificador são activas a 0, e estão desactivadas a não ser que um endereço seleccione uma das saídas.

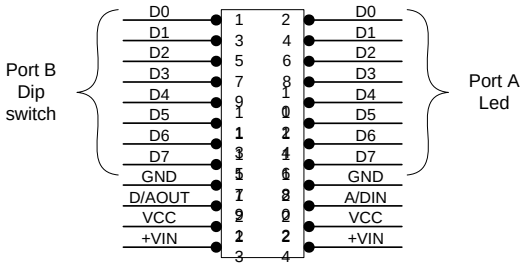
A selecção da EPROM é válida para os endereços de memória 0000H a 3FFFH ou 7FFFH e é feita através do seu enable. A selecção da RAM é válida para os endereços 4000H a BFFFH ou 8000H a FFFFH, o que equivale a um intervalo de 32K. Como os descodificadores de memória descodificam intervalos de 16K, os

dois circuitos seleccionados são lógica negativa. Parte da memória do circuito seleccionado é devolvida no intervalo C000H a FFFFH. A parte I/O existe nos endereços 00H a 3FH.





Entradas e Saídas

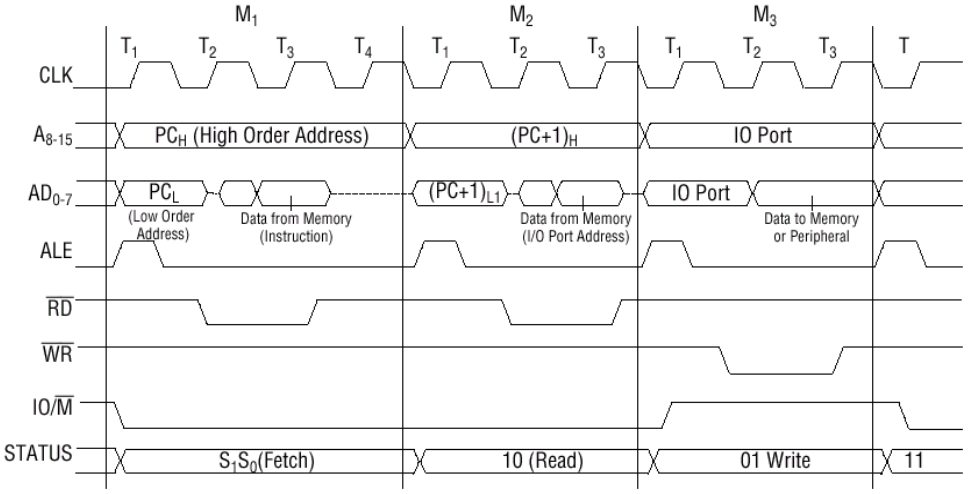


	Descrição	Ender.	Função
8251	Registo de dados	80H	Input / output
8251	Registo de Control	81H	Configuração
8155	Registo de Control	10H	Configuração
Port A		11H	output Led
Port B		12H	input Dipswitch
Port C		13H	analog output
Timer low		14H	
Timer high		15H	
Exp. I/O		C0-FFH	

Registo de Control do 8155	
bit 0	Port 11 Hex: 0 – Output , 1 - Input
bit 1	Port 12 Hex: 0 – Output , 1 - Input
bit 2	
bit 3	
bit 4	
bit 5	
bit 6	
bit 7	

Timing do sistema

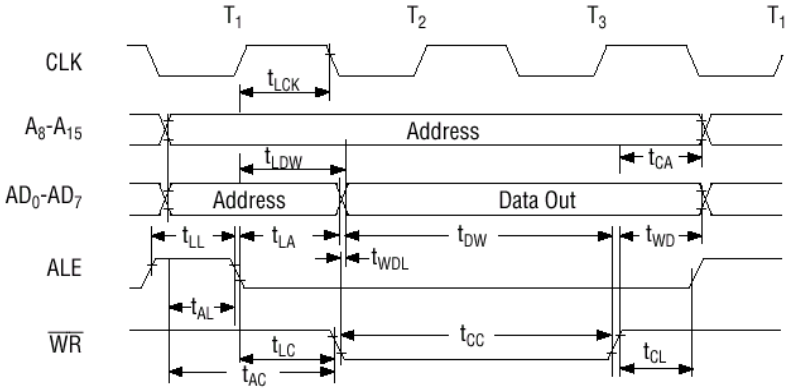
O 8085 tem um Bus de dados multiplexado. O ALE é usado para validar os 8 bits de endereço menos significativos no Bus de Dados. A figura seguinte mostra a actividade no Bus do 8085 durante a busca de uma instrução (fetch) e escrita em I/O (input/output). Note que durante o ciclo de acesso a I/O, o endereço de I/O é colocado tanto nos 8 bits de endereço mais significativos como nos 8 bits de endereço menos significativos.



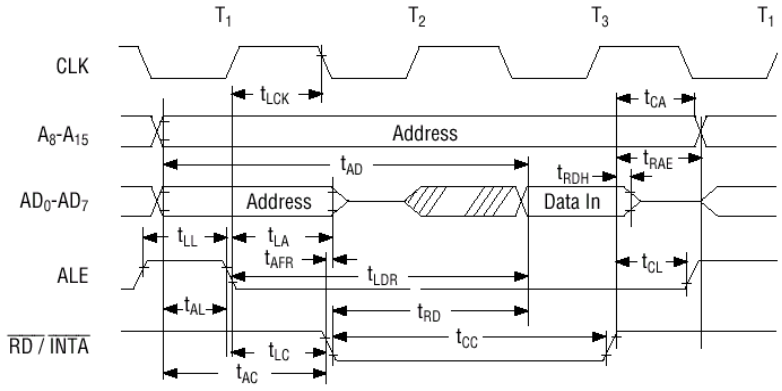
80C85 Basic System Timing

Nas figuras seguintes estão os sinais do 8085 referente a um ciclo de leitura e escrita de memória.

WRITE OPERATION



READ OPERATION



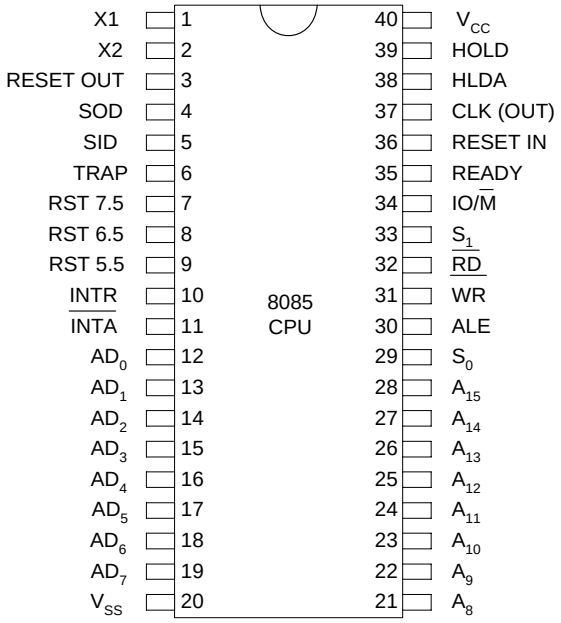
MICROPROCESSADORES

O 8085 é o cérebro do sistema. Ele coordena quase toda a actividade. O 8085 é constituído por um conjunto de contadores, portas lógicas, registos, descodificadores, etc, que sequencialmente retiram instruções da memória, descobrem a sua finalidade e executam-nas uma a uma. As instruções são colocadas na memória na ordem em que vão ser executadas. Os tipos de instruções são extremamente variadas, mas muito concisas. As suas operações consistem em deslocar dados, saltos incondicionais e condicionais, algumas operações matemáticas, e Input/Output. Quando correctamente juntas numa ordem lógica, estas instruções primitivas formam aquilo a que se chama programa. Os programas podem simplesmente deslocar dados de uma porta de entrada para uma porta de saída, usando algumas instruções, ou podem proporcionar operações complexas de controlo usando ports de entrada e saída e um número de instruções de vários milhares.

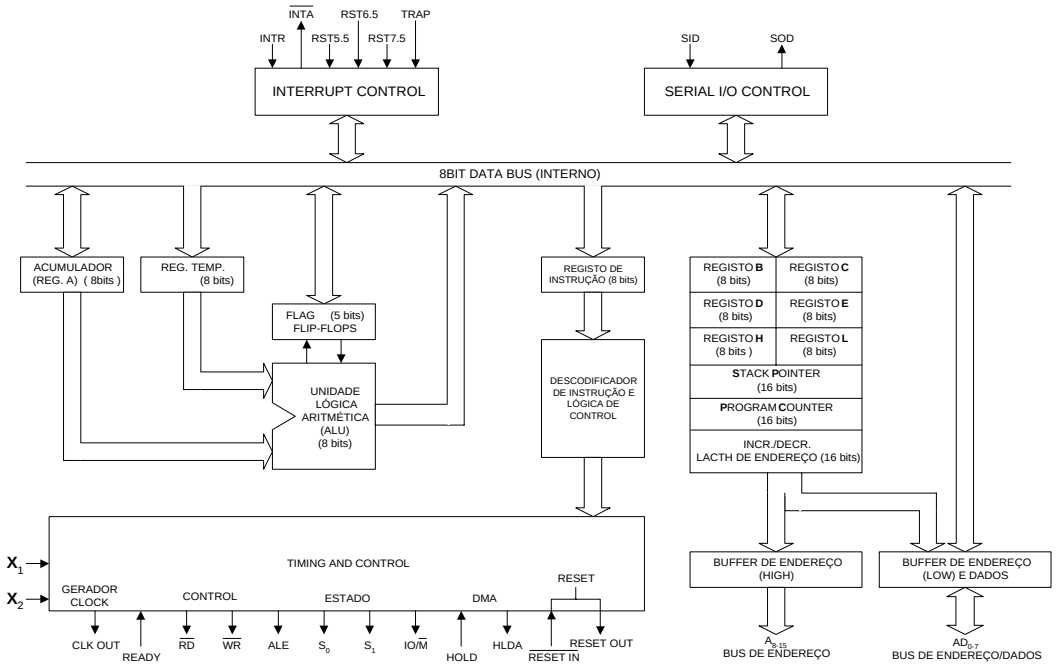
O 8085 foi construído para poder aceder a 65536 posições possíveis de memória e 256 posições I/O. A arquitectura do microprocessador espera que o programa e os dados estejam residentes de uma forma sequencialmente ordenada na memória, para poder operar correctamente. O programador deve escrever o programa correctamente de maneira que o microprocessador saiba quais os conteúdos da memória que são instruções e os que são dados.

O 8085 e muitos outros microprocessadores começam a executar o programa desde a primeira posição de memória (0000 Hex), logo após ter sido ligado ou após um reset. Isto é accionado por um circuito de reset quando o sistema é ligado ou se pressiona o botão de reset.

Externamente o 8085 apresenta-se igual a qualquer circuito integrado com 40 pinos, uma caixa preta com duas filas laterais de pinos de acordo com a figura seguinte.



Além da estrutura lógica necessária para executar todas as funções que são definidas pelo seu conjunto de instruções (instruction set), o microprocessador 8085 possui ainda, no seu interior, a lógica necessária à execução das funções de um gerador de impulsos de relógio, de um controlador do Bus do sistema e de um seleccionador de prioridades das interrupções do sistema.



UNIDADE DE CONTROLO E DESCODIFICADOR DE INSTRUÇÕES

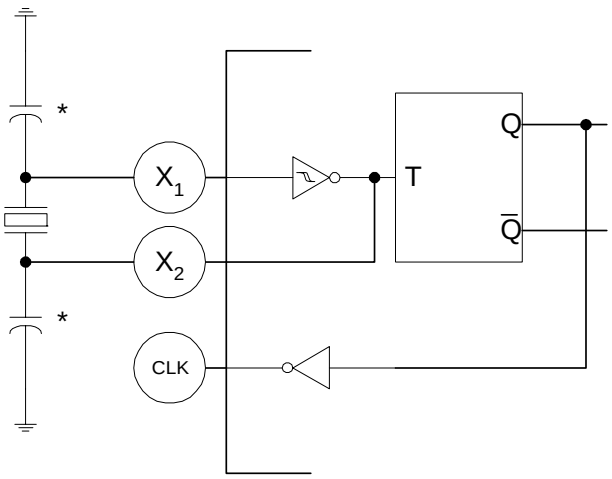
As funções da lógica de controlo e temporização são: descodificar a instrução (descodificador de instruções), gerar sinais de temporização (oscilador interno de relógio) e combinar os sinais obtidos da descodificação com os sinais de temporização afim de serem de serem obtidos os sinais de controlo que serão enviados às diferentes partes do 8085 e do sistema.

CLOCK

A figura seguinte apresenta-nos a estrutura lógica do temporizador interno do 8085. O 8085 utiliza um Schmitt Trigger quer como oscilador quer como condicionador de entrada., dependendo do facto de se utilizar na entrada x1, x2 um cristal ou uma fonte externa de frequência. Este circuito de temporização gera dois sinais de relógio internos sobrepostos, ∅1 e ∅2, que sincronizam o funcionamento interno do 8085 e produzem o sinal de saída de CLK.

X_1, X_2 (Entrada, Saída). Estes pinos são ligados a um cristal ou a um circuito RC ou LC, afim de realimentar o circuito interno e provocar a oscilação. X_1 também pode ser uma entrada de um gerador de clock externo. A frequência de entrada é dividida por 2 para se obter a frequência interna de funcionamento do microprocessador.

CLK (Saída). Este pino suporta uma saída de impulsos de clock para temporização do sistema. O período à saída de CLK é duas vezes maior que o período do sinal de entrada X_1.



*- Condensadores necessários apenas no caso de o cristal ter uma frequência≤ 4 MHz

REGISTOS

Este grupo de registos suporta a ligação do microprocessador com o mundo exterior, e é composto por:

Registo de endereços (Buffer de endereço); este registo é usado para assegurar a informação a transmitir pelos pinos A8-A15 que é o Byte mais significativo do endereço.

Registo de dados/Endereço (Buffer de endereço e dados); este é o registo que apresenta dupla função de assegurar a transmissão do byte menos significativo do endereço e assegurar a transmissão do dado através dos pinos AD0-AD7.

Registo de controlo de interrupções (Interrupt control); este registo recebe os pedidos de interrupção, que lhe chegam pelos pinos INTR, RST5.5, RST 6.5, RST 7.5 e Trap. Controla a prioridade e envia o sinal INTA quando é caso disso.

Registo de controlo de E/S série (Serial I/O control); este registo destina-se a suportar as funções para os pinos SID e SOD.

Seis registos de uso genérico; estes registos são de oito bits e organizados aos pares, sendo cada um desses pares utilizado, para algumas funções, como se fosse um único registo de 16 bits. Quando funcionam como registos simples de 8 bits, as designações que tomam são as que lhes são atribuídas na figura anterior (B, C, D, E, H, L). Quando são utilizados como registos de 16 bits as designações são as seguintes:

B para o par BC

D para o par DE

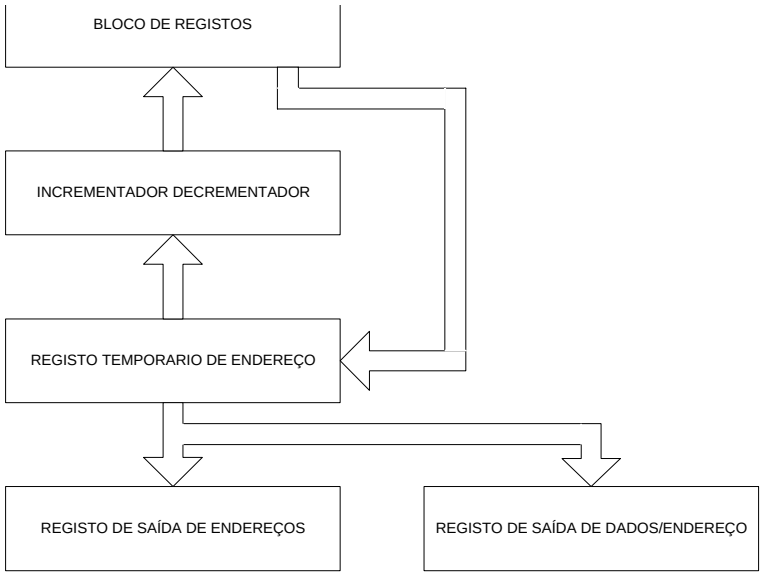
H para o par HL (usado como ponteiro de memória)

Stack Pointer (Ponteiro de pilha); este registo permite a organização de uma pilha de informação em qualquer zona de memória, pelo programador ou pelo programa monitor, para nela guardar a informação contida no acumulador ou/e nos registos, afim de que não seja destruída quando se chama uma sub rotina. O nome pilha é devido ao processo de introdução e extracção da informação: quando se introduz informação na pilha, o ponteiro de pilha é decrementado, de modo que a informação a extrair terá endereço numericamente superior ao da informação que foi extraída anteriormente. Por outras palavras: a última informação a ser introduzida é a primeira a sair e vice-versa.

Program Counter (Contador de programa); é um registo que contém o endereço da instrução a executar e é incrementado de uma unidade sempre que se quer passar à execução da instrução seguinte.

Incrementador e decrementador; este bloco é na realidade, algo mais do que um registo. A figura seguinte

pretende, através de um diagrama de blocos, dar uma ideia do que se passa nesta unidade. Os endereços são fornecidos ao registo temporário de endereços por um qualquer dos pares do bloco de registos. O registo temporário fornece esses endereços aos registos de saída de endereços (Adress Buffer e Data/Adress Buffer) ou ao incrementador/decrementador, conforme o comando que recebe da unidade de controlo.



UNIDADE LÓGICA E ARITMÉTICA

Unidade Lógica e Aritmética; Esta unidade não constitui já novidade para nós, tal como as operações aritméticas e lógicas que executa. No entanto, não dissemos ainda que a operação de deslocamento dos bits de uma palavra, era outra das funções executadas por esta unidade.

Os registos associados à ALU são:

O acumulador

O registo temporário

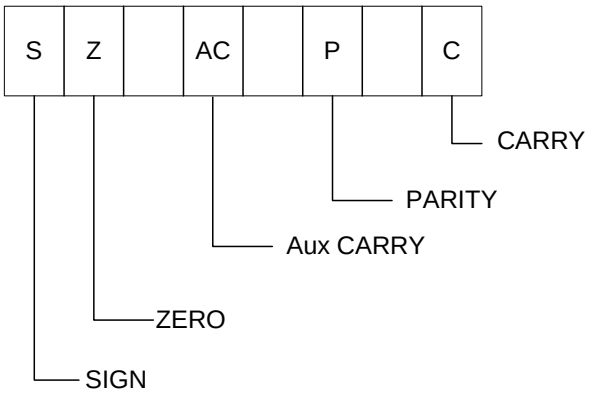
O registo de Flags

Já conhecemos a importância do acumulador e o contacto constante que iremos ter com ele pela quantidade de vezes que é requisitado pelo repertório de instruções, levar-nos-á decerto, ao conhecimento profundo deste registo privilegiado.

A respeito do registo temporário não há muito a dizer, pois se apresenta obvia a função que desempenha junto à aritmética e lógica.

Iremos nos debruçar um pouco mais sobre o registo de Flags, dada a sua importância no desenvolvimento da programação do 8085. Comecemos por dizer que é constituído por um conjunto de flip-flops que apresentam indicação de determinadas condições, quando finalizado o processo de execução de cada operação. Por este motivo são designados por Flags. A constituição deste registo, bem como o símbolo e designação de cada flip-flop (Flag) encontra-se na figura seguinte.

A Flag CARRY dá-nos a indicação da ocorrência de uma situação de transporte a partir do bit mais significativo do acumulador; isto é, quando o resultado de uma operação ultrapassa a capacidade do acumulador (8 bits) verifica-se uma activação da Flag CARRY (nível lógico 1).



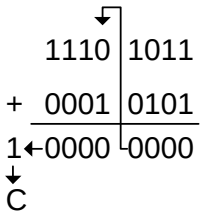
A Flag PARITY indica-nos qual a paridade (paridade par => P=1 e paridade impar => P=0) do conteúdo do acumulador.

A Flag AUXILIARY CARRY dá indicação da existência de transporte a partir do bit 3 do acumulador tal como a flag CARRY dava no caso do bit 7. Esta Flag encontra grande aplicação nas operações com BCD.

A Flag ZERO informa se o resultado da última operação da ALU deu ou não um resultado zero (no caso de ser zero => Z=1).

A Flag SIGN indica qual o valor lógico do bit 7 do acumulador. Como o bit 7 é, por vezes, reservado para o sinal, daí a designação SIGN (sinal) para esta Flag (bit 7 = 0 => S = 0, bit 7 = 1 => S = 1).

O exemplo que a seguir se apresenta, serve para nos dar uma ilustração prática dos valores das flags após a soma de dois operandos.



CARRY = 1

PARITY = 1 – Não existem uns, logo a paridade é par

AUXILIARY CARRY = 1 – há transporte a partir do bit 3

ZERO = 1 - O conteúdo do acumulador é zero

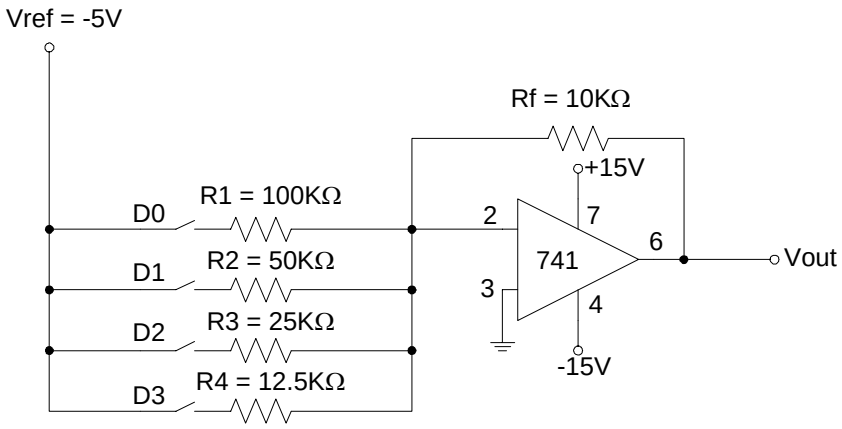
SIGN = 0 – o bit 7 tem o valor lógico 0

O registo de Flags é utilizado na programação do 8085 para permitir efectuar rotinas com decisões. O estudo das instruções do 8085, que utilizam as Flags deste registo, irão dar-nos a oportunidade de melhor nos familiarizarmos com ele e tirarmos o máximo rendimento das informações que fornece.

CONVERSÃO DE DADOS

CONVERSORES DIGITAIS ANALÓGICOS

A função do conversor digital analógico é converter uma palavra em binário para uma tensão ou corrente proporcional ao seu valor. Para vermos como isto é feito vamos analisar o circuito da figura seguinte.



Uma vez que a entrada não inversora está ligada à massa, o amplificador vai colocar uma tensão no pino 6 de modo que a tensão na entrada inversora seja sempre igual a 0V. Quando um dos interruptores é ligado, a corrente vai fluir dos –5V através da resistência até à entrada não inversora. Para que a tensão nesta entrada seja 0V é necessário que esta corrente flua toda pela resistência Rf.

Se por exemplo fechar o interruptor D0, uma corrente de 0.05mA vai circular pela resistência R1. Para que esta corrente flua toda pela resistência Rf, é necessário por uma tensão de 0.05mA x 10KΩ ou 0.5V na saída do amplificador. Se fecha também o interruptor D1, vai circular mais 0.1mA. Para que as duas correntes (0.05mA + 0.1mA) circulem pela resistência Rf, o amplificador tem que ter a saída uma tensão igual a 0.15mA x 10KΩ ou 1.5V.

As resistências produzem uma corrente com um peso igual ao peso em binário do respectivo interruptor, são somadas pelo amplificador para produzir uma tensão que é proporcional. A palavra em binário colocada nos interruptores produz uma tensão à saída proporcional. Tecnicamente a tensão à saída é digital porque só pode ter alguns valores fixos, tal como o display de um voltímetro digital. Entretanto, a saída simula um sinal analógico, por isso dizemos que é analógico.

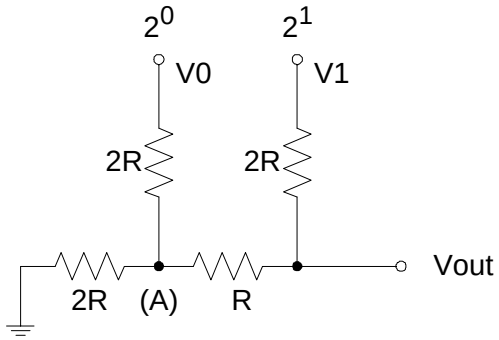
O interruptor D3 representa o bit mais significativo porque quando é accionado produz a maior corrente. Note que Vref é negativo, mas a saída é positiva.

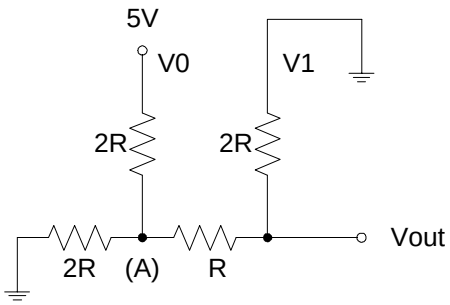
CONVERSOR DIGITAL ANALÓGICO R-2R

Na figura seguinte podemos ver um exemplo de um conversor digital analógico de 2 bits que utiliza uma malha de resistências.

Vamos fazer uma análise deste circuito com auxílio da tabela de verdade.

V1	V0	Vout
0V	0V	0V
0V	5V	1.25V
5V	0V	2.5V
5V	5V	3.75V





Na primeira condição, quando ambas as entradas estão ligadas a 0V, não existe corrente a circular no circuito e por isso não há queda de tensão. Vout é igual a 0V.

Vamos agora analisar a segunda condição, V1 = 0V e V0 = 5V. Em primeiro lugar vamos calcular a tensão no ponto (A).

Calcular a resistência equivalente entre o ponto (A) e a massa:

$$R(A) = \frac{2R \times 3R}{(2+3)R} = \frac{6}{5}R$$

Calcular a tensão no ponto (A):

$$V(A) = 5V \text{ x } \frac{6/5 R}{2R + 6/5 R} = 5V \text{ x } \frac{6}{16} = 1.875V$$

Podemos agora calcular Vout do mesmo modo que foi calculado V(A):

$$Vout = 1.875V \text{ x } \frac{2R}{3R} = 1.875V \text{ x } \frac{2}{3} = 1.25V \;\; ; \;\; Vout = 5V \text{ x } \frac{6}{16} \text{ x } \frac{2}{3} = 5V \text{ x } \frac{12}{48} = 5V \text{ x } \frac{1}{4} = 1.25V$$

Todas as outras tensões podem ser analisadas da mesma forma. Verifique que a tensão aumenta em passos de 1.25V. A ultima condição da tabela é igual a 3.75V. Falta sempre mais um passo no final da tabela para atingir a tensão máxima (3.75V + 1.25V = 5V), qualquer que seja o número de bits usados.

A fórmula simplificada usada para calcular a tensão de saída é:

$$Vout = \frac{V1}{2} + \frac{V0}{4}$$

CONVERSOR R-2R DE 8 BITS

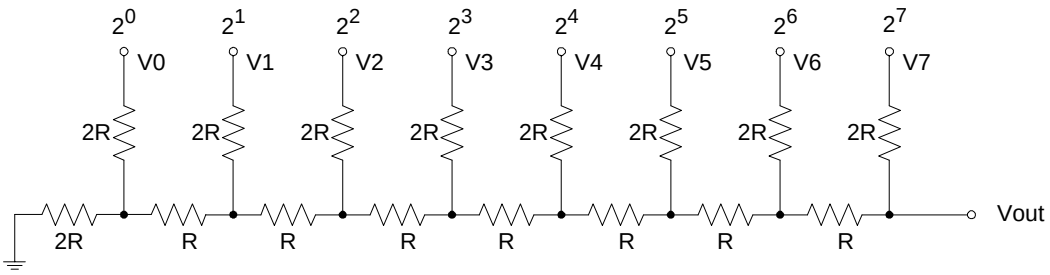
Este conversor é igual ao anterior com a vantagem de haver maior número de bits. A fórmula usada para obter a tensão de saída é:

$$Vout = \frac{V7}{2} + \frac{V6}{4} + \frac{V5}{8} + \frac{V4}{16} + \frac{V3}{32} + \frac{V2}{64} + \frac{V1}{128} + \frac{V0}{256}$$

Uma das primeiras características de um conversor é a sua resolução. Isto é determinado pelo número de bits. Um conversor com 8 bits como este tem 2^8 ou 256 níveis de saída. O exemplo do circuito anterior tem 2 bits e por isso apenas 4 níveis de saída. A resolução também pode ser expressa em percentagem. A resolução de um conversor de 2 bits é de $1/4$ x 100% ou de 25%. A resolução de um conversor de 8 bits é de $1/256$ x 100% ou de 0.39%.

A próxima característica é a tensão máxima que se pode obter à saída do conversor. Quanto maior o número de bits, mais próxima é a tensão máxima da tensão digital.

$$Vmax = \frac{2^{n^o \text{ bits}} - 1}{2^{n^o \text{ bits}}} = \frac{255}{256} = 0.996 \text{ x } 5V = 4.98V$$



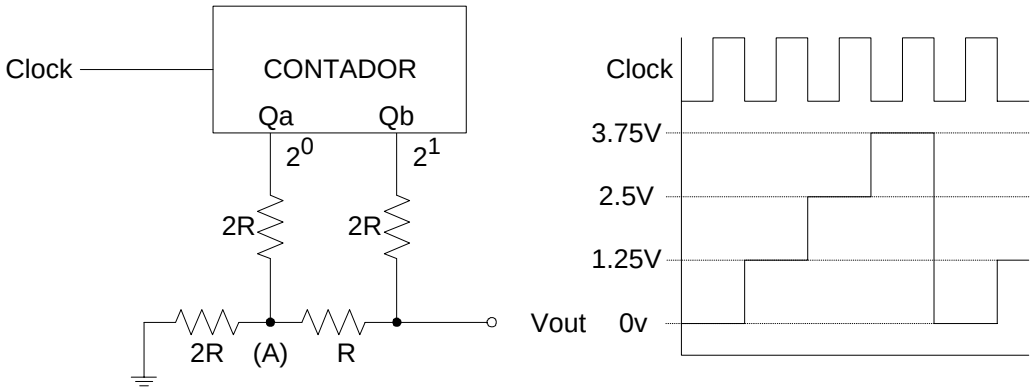
Existe outra característica importante de um conversor D/A que é a linearidade. A linearidade é uma medida de quanto a saída do conversor se desvia de uma linha recta à medida que a tensão do conversor progride. Idealmente, o desvio entre a saída do conversor e uma linha não recta não deve ser maior que $\pm \frac{1}{2}$ do valor do bit menos significativo. Entretanto, muitos conversores D/A têm erros de linearidade maiores que esse valor.

Ainda existe outra característica para ver que é o tempo de atraso. Quando se altera a palavra em binário aplicada na entrada do conversor, a saída vai variar para o valor apropriado. Entretanto a saída só vai assumir esse valor passado algum tempo. O tempo que leva a variar a saída $\pm \frac{1}{2}$ do bit menos significativo chama-se tempo de atraso do conversor. Por exemplo, o conversor de 10 bits DAC1020 tem um tempo de atraso típico de 500ns. Esta característica é importante quando um conversor é construído para trabalhar em alta frequência.

Aplicações do conversor D/A

Estes conversores têm muitas aplicações, além disso são muito usados nos computadores e leitores de Compact Disk. No leitor de CD, são usados conversores D/A de 14 ou 16 bits para converter os dados lidos do disco num sinal áudio. A maior parte dos sintetizadores de voz contêm um conversor D/A.

Vamos utilizar um conversor D/A para ligar na saída dum contador de 2 bits.



Verifique que a tensão de saída é semelhante a uma onda dente de serra em escada. Cada degrau é igual

ao peso do bit menos significativo e também igual a $Vout = 5V \times \frac{1}{4} = 1.25V$ como já verificado anteriormente. O tempo de cada degrau corresponde também à duração de um ciclo completo do Clock. Quanto maior número de bits do contador e do conversor maior será o número de degraus entre o valor mínimo e máximo. Isto significa uma melhor resolução da onda dente de serra.

CONVERSOR ANALÓGICOS DIGITAIS

A função do conversor analógico digital A/D é produzir uma palavra digital que representa a magnitude de um valor analógico. As especificações de um conversor A/D são muito semelhantes com as do conversor D/A. A resolução de um conversor A/D refere-se ao número de bits na palavra digital à saída do conversor. Um conversor A/D de 8 bits, por exemplo, tem uma resolução de $1/256$. A precisão e linearidade têm o mesmo significado no conversor A/D que no conversor D/A.

Uma outra característica importante num conversor A/D é o tempo de conversão. Isto é simplesmente o tempo que o conversor leva para produzir uma saída em binário para um valor analógico aplicado na entrada. Quando nos referimos a um conversor de alta velocidade, queremos dizer que ele tem um tempo de conversão muito pequeno.

Existem várias maneiras para fazer uma conversão A/D, vamos analisar alguns dos métodos mais usados.

TIPOS DE CONVERSORES A/D

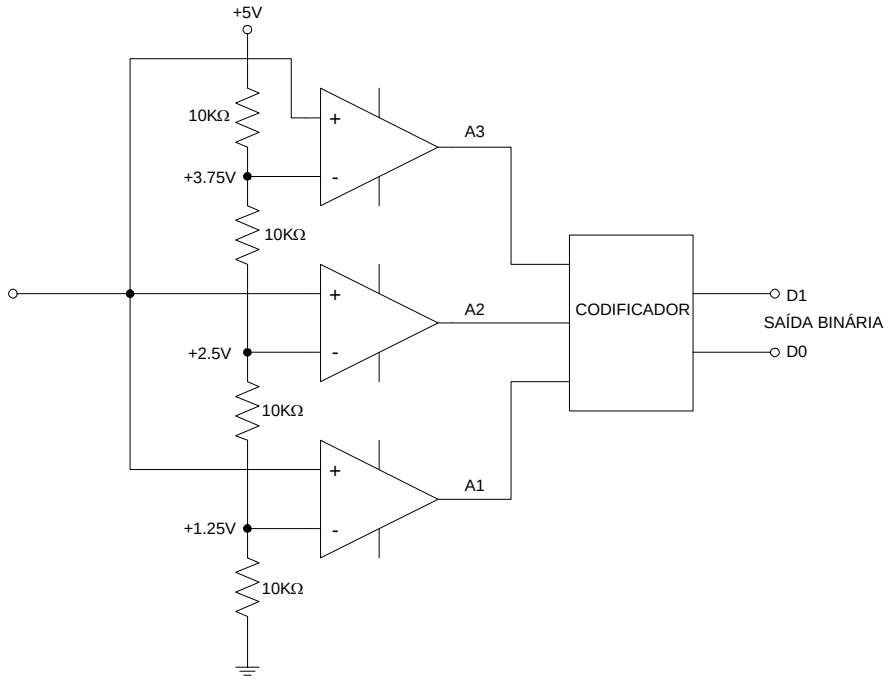
Conversor com comparador paralelo

A figura seguinte mostra um circuito conversor A/D de 2 bits usando comparadores. Um divisor de tensão fornece as tensões de referência na entrada de cada um dos comparadores. A tensão no topo do divisor representa a tensão do nível lógico 1. Se a tensão na entrada de um comparador é superior à tensão de referencia na entrada não inversora, a saída do comparador passa a nível lógico 1. As saídas dos comparadores dão-nos uma representação digital da tensão de entrada. Com uma tensão de entrada de 2.6V, por exemplo, as saídas dos comparadores A1 e A2 vão estar a nível lógico 1.

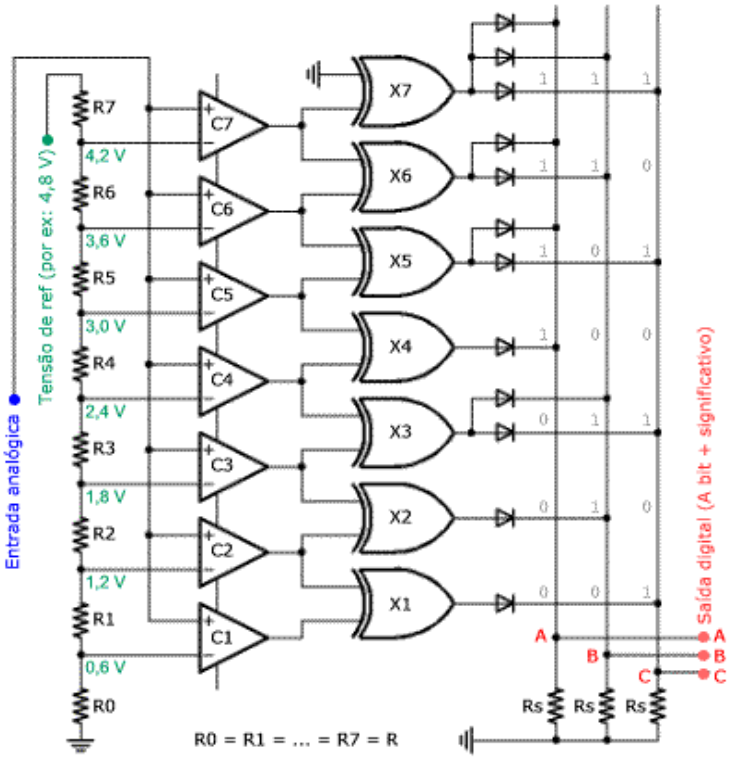
A maior vantagem do conversor A/D paralelo é a sua velocidade, que é simplesmente o tempo de atraso dos comparadores. A saída dos comparadores é um formato binário não standard, mas pode ser convertido para qualquer código binário desejado com algumas portas lógicas. A maior desvantagem de um conversor A/D paralelo é o número de comparadores necessários para produzir um resultado com uma resolução razoável.

O conversor de 2 bits da figura seguinte requer 3 comparadores. Para construir um conversor de N bits precisa de $2^N - 1$ comparadores. Para um conversor de 8 bits, precisa de 255 comparadores, e para um

conversor de 10 bits precisa de 1023 comparadores. Existem conversores A/D com comparadores paralelos em circuitos integrados para serem usados em aplicações onde a alta velocidade é necessária, mas são relativamente caros. Alguns conversores com comparadores paralelos podem fazer uma conversão em menos de 10ns.

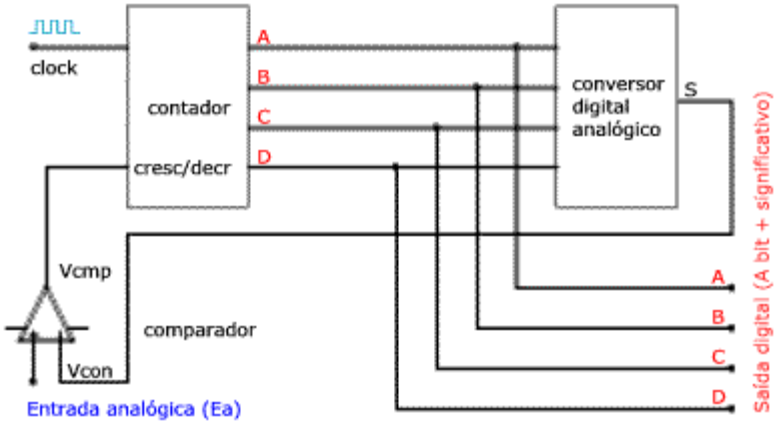


Conversor de 3 bits:



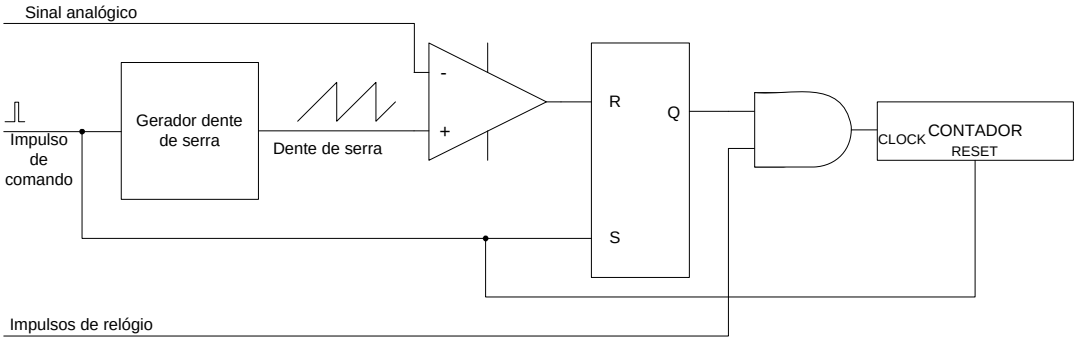
No Conversor Delta , os impulsos de clock alimentam continuamente a entrada do contador, o qual dispõe de uma entrada digital que comuta, de acordo com o nível lógico, o sentido da contagem (crescente ou decrescente). Enquanto a entrada analógica é maior que Vcon, a saída do comparador é 1 e o contador funciona no modo crescente. Quando Vcon se torna maior que “Ea”, a saída do comparador vai para 0 e o contador funciona no modo decrescente.

Isso leva Vcon a um valor imediatamente abaixo de “Ea”, invertendo o processo. Assim, considerando “Ea” constante, o contador opera continuamente entre dois valores próximos de “Ea”, não havendo necessidade dos flip-flops de armazenamento. Se o valor de “Ea” muda, o patamar de operação também muda.



Conversor de rampa simples

Este tipo de conversor é bastante vulgar, o esquema é apresentado na figura seguinte. Um impulso de comando dá inicio à onda dente de serra, acciona o flip-flop RS e acciona o RESET do contador. A partir deste momento a saída do flip-flop é 1 e é aplicada na porta AND. O contador começa a contar a partir do zero e a tensão da onda dente de serra começa a subir a partir do zero. Enquanto a tensão de dente de serra for inferior à tensão do sinal analógico, a contagem continua, mas quando a tensão dente de serra passa pela tensão analógica o comparador comuta e acciona a entrada R do flip-flop. Isto desactiva a porta AND e termina assim a contagem. O número da contagem é assim proporcional à tensão analógica. O impulso de comando seguinte coloca o contador na posição inicial, a dente de serra a zero e começa uma nova contagem.



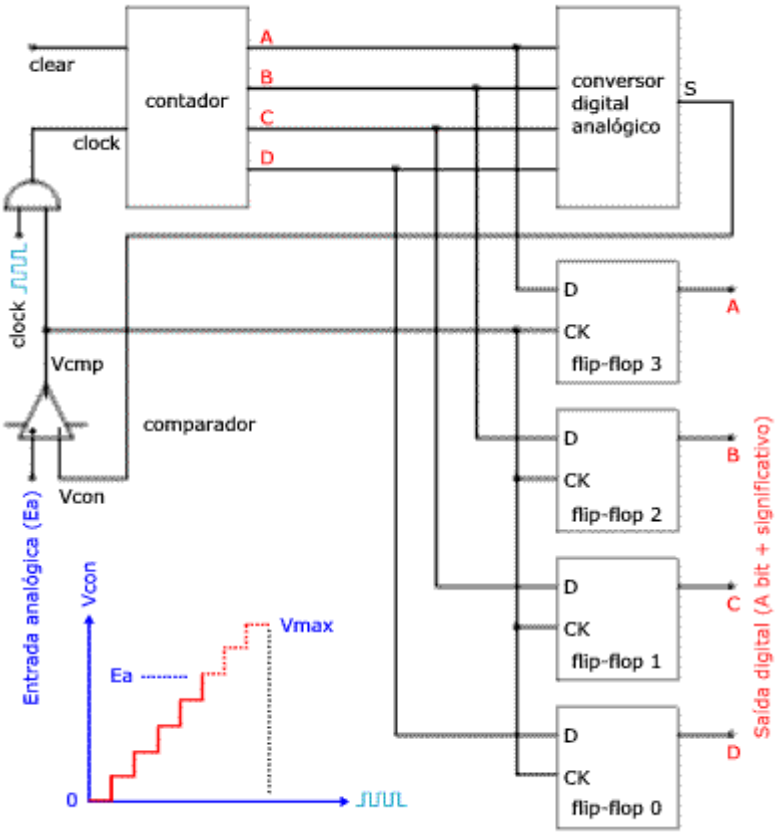
Este conversor também pode ser feito da seguinte forma; a saída de um contador (de 4 bits neste exemplo) é ligada na entrada de um conversor digital analógico. Supomos de início que a entrada de clock do contador é continuamente alimentada com uma sequência de impulsos. Nesta situação, a tensão Vcon na saída S do conversor varia entre 0 e um valor Vmax, que depende do contador e das características do conversor digital analógico.

Mas no circuito há um comparador e uma porta AND na entrada de clock. Enquanto a tensão Vcon for menor que a da entrada analógica “Ea”, a saída do comparador é 1 e os impulsos de clock são dirigidos ao contador.

No momento em que Vcon se torna maior que “Ea”, a saída do comparador passa para 0, bloqueando os impulsos de clock e, portanto, a contagem.

Uma vez que a saída do comparador também vai para a entrada de clock dos flip-flops (tipo D), o valor

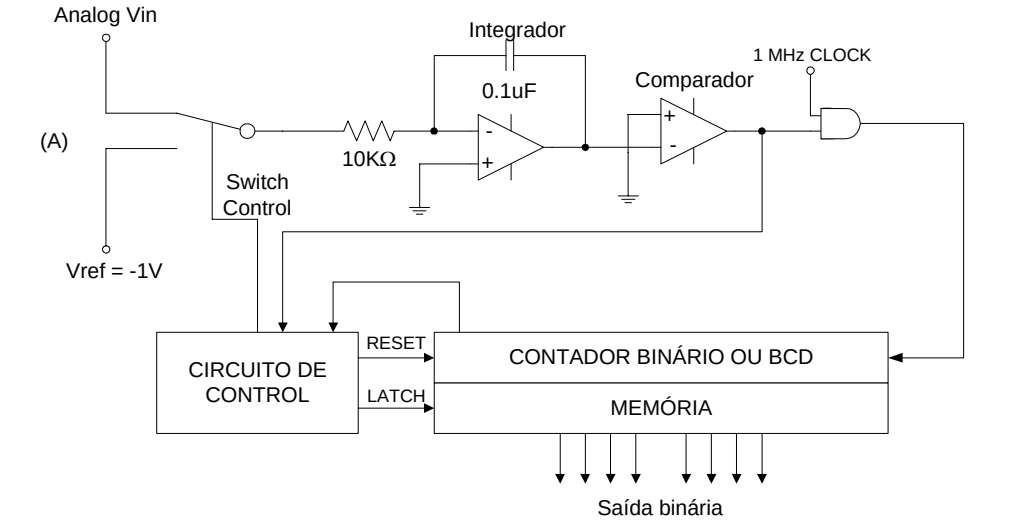
digital da saída do contador é armazenado nos mesmos (lembrar que flip-flops tipo D só permitem a mudança de estado na transição de clock, neste caso descendente). Portanto, a saída digital armazenada nos flip-flops tem relação linear com a entrada analógica “Ea”.

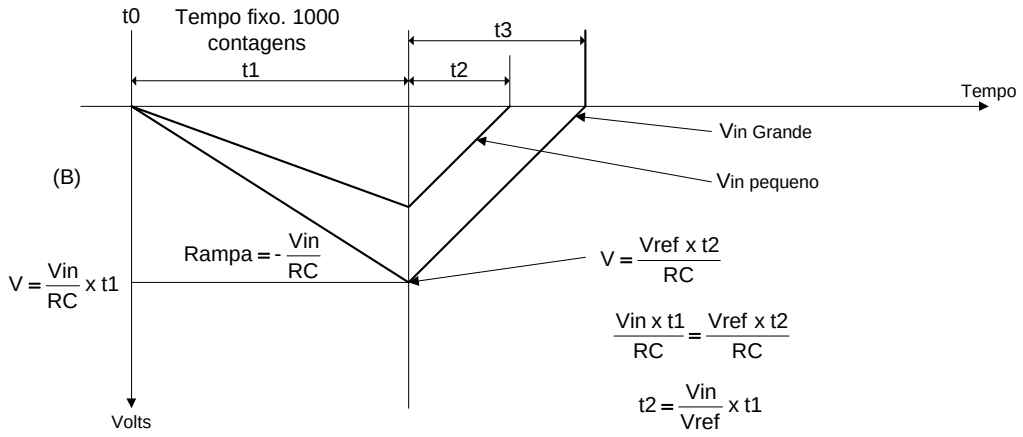


Conversor A/D de dupla rampa

A figura seguinte (A) mostra o esquema bloco de um conversor A/D de dupla rampa. Este tipo de conversor é normalmente usado nos multímetros digitais porque pode fornecer um grande numero de bits de

resolução a baixo custo. Vamos ver como ele funciona.



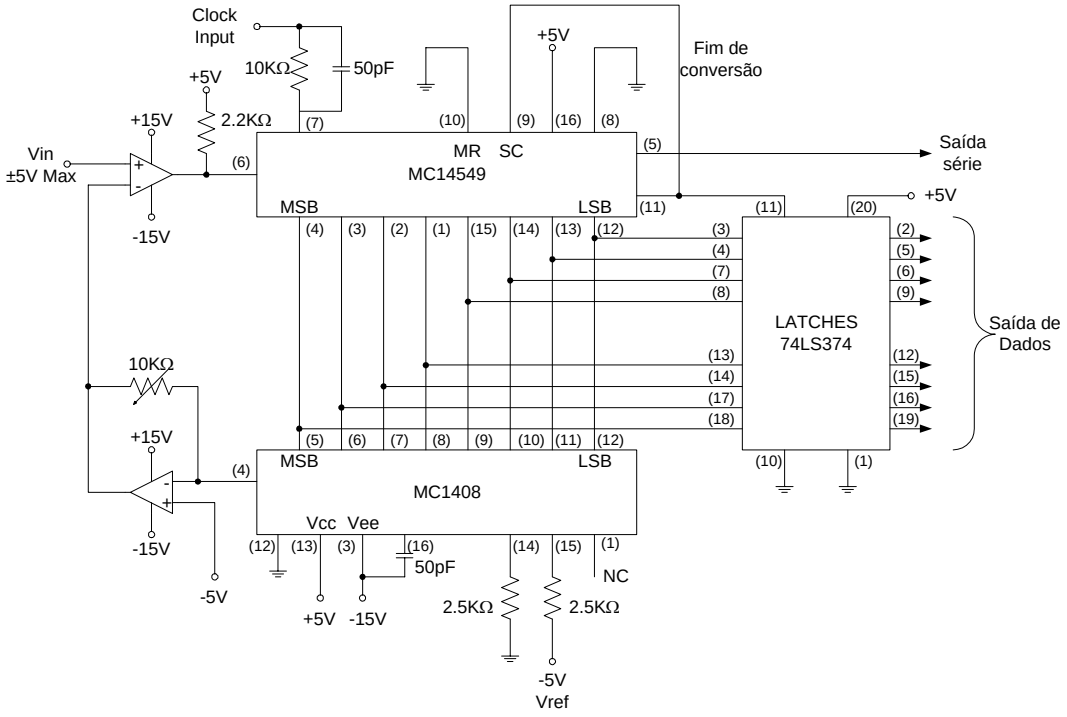


Para começar, o circuito de controlo coloca todos os contadores a zero e liga a entrada do integrador à tensão de entrada analógica. Se assumir que a tensão de entrada é positiva, então a saída do integrador descerá para negativo como está representado em (B) na figura anterior. Assim que a saída do integrador desça alguns microvolts abaixo de zero, a saída do comparador passará a nível lógico 1. A saída do comparador a 1 acciona a porta AND e permite que o clock de 1 MHz chegue ao contador. Normalmente após 1000 impulsos contados no contador, o circuito de controlo comuta a entrada do integrador para a tensão de referencia negativa e coloca todos os contadores a zero. Com uma tensão de entrada negativa, a saída do integrador subirá para positiva como se mostra na direita da figura anterior (B). Quando a saída do integrador passa por zero, a saída do comparador passa a nível lógico 0 e desliga o sinal de 1 MHz para os contadores. O numero de contagens necessárias para a tensão de saída do integrador voltar a zero é directamente proporcional à tensão de entrada. Por exemplo, uma tensão na entrada de +2V produz uma contagem de 2000. Uma vez que a resistência e o condensador no integrador são usados tanto pela tensão de entrada como pela tensão de referencia, pequenas variações nos seus valores devido às variações de temperatura não tem efeito na precisão da conversão.

A principal desvantagem dum conversor A/D de dupla rampa é a sua baixa velocidade de conversão.

Conversor A/D de aproximações sucessivas

A figura seguinte mostra um conversor A/D de aproximações sucessivas de 8 bits. O coração deste conversor é um registo de aproximações sucessivas como por exemplo o MC14549, que funciona da seguinte maneira.



No primeiro impulso de clock do início de um ciclo de conversão, o MC14549 coloca a saída do bit mais significativo a 1 para o conversor D/A MC1408. O conversor D/A e o amplificador convertem este bit para uma tensão que é aplicada na entrada do comparador. Se esta tensão é maior que a tensão de entrada na outra entrada do comparador, a saída do comparador passa a lógico 0 e avisa o MC14549 para desligar este bit (MSB) porque é muito grande. Se a tensão do conversor D/A é inferior à tensão de entrada, a saída do comparador passa a lógico 1, e avisa o MC14549 que deve manter este bit (MSB) ligado. No próximo impulso de clock, o MC14549 vai ligar o próximo bit a seguir ao MSB para o conversor D/A. Baseado na resposta obtida no comparador, MC14549 irá manter ou desligar este bit. O MC14549 prossegue assim até ao bit menos significativo. Quando a conversão estiver completa, o resultado em binário está presente na saída do MC14549, e o MC14549 envia um sinal de fim conversão. Apenas são necessários nove impulsos de clock para efectuar toda a conversão. No esquema anterior, o sinal de fim de conversão é usado para memorizar a informação nas latches, assim a informação pode ser usada para ser lida por um micro-computador. Se a saída de fim de conversão é ligada na entrada SC (inicio de conversão), então o conversor vai efectuar conversões sucessivamente.

BIBLIOGRAFIA

Padilla, António J. G .-Sistemas Digitais.

Hall, Douglas V. Hall-Microprocessors and Interfacing Programming and Hardware

Matemática II

Apontamentos Pessoais

LISTA DE PÁGINAS EM VIGOR

PÁGINAS	EM VIGOR
CAPA (Verso em branco)	ORIGINAL
CARTA DE PROMULGAÇÃO (Verso em branco)	ORIGINAL
REGISTO DE ALTERAÇÕES (Verso em branco)	ORIGINAL
1 (Verso em branco)	ORIGINAL
3 a 16	ORIGINAL
17(Verso em branco)	ORIGINAL
19 a 38	ORIGINAL
39 (Verso em branco)	ORIGINAL
41 a 42	ORIGINAL
43 (Verso em branco)	ORIGINAL
45 a 92	ORIGINAL
93 (Verso em branco)	ORIGINAL
95 a 104	ORIGINAL
105 (Verso em branco)	ORIGINAL
107 (Verso em branco)	ORIGINAL
LPV-1 (Verso em branco)	ORIGINAL