Universidade de Alfenas
Bacharelado em Ciência da Computação

NOTAS DE AULA

ARQUITETURA DE COMPUTADORES
Microarquitetura

Prof. Marcos Alberto de Carvalho

MICROARQUITETURA E MICROPROGRAMAÇÃO

1. INTRODUÇÃO

Uma característica extremamente fascinante nos microprocessadores é a sua capacidade de interpretar e executar instruções de programa.

Na arquitetura de qualquer sistema computacional, o microprocessador é o elemento que define a potencialidade da máquina, a partir de sua velocidade de processamento, o tamanho da palavra manipulada, a quantidade de memória interna (registros) e seu conjunto de instruções.

A maior parte destes fatores está intimamente ligada à forma como os diferentes componentes do microprocessador estão associados e como estes são controlados internamente. A esta organização é dado o nome de **microarquitetura**.

Cada microprocessador possui sua microarquitetura específica e, como consequência disto, sua própria linguagem de máquina.

O objetivo deste capítulo é introduzir o conceito de microarquitetura e de microprogramação como fatores determinantes dos aspectos de execução de um microprocessador.

Dada a grande diversidade de configurações em termos de microarquitetura e a grande complexidade dos fatores associados a ela, será feito uso de um exemplo didático, mas bastante representativo de alguns casos reais.

2. OS MICROPROCESSADORES E OS SINAIS EXTERNOS

Os microprocessadores são geralmente circuitos integrados dispostos em pastilhas onde o número de pinos varia de 40 a 132. Como ilustrado na figura 3.1, os sinais associados a estes pinos permitirão ao microprocessador a troca de informação com o seu ambiente, ou seja, memória e circuitos de E/S.

Uma análise detalhada dos sinais disponíveis nos pinos de um microprocessador permite conhecer os seus diferentes modos de funcionamento (lógico e elétrico) e as suas possibilidades em termos de interfaces.

Os sinais externos dos computadores são organizados em três grupos: os sinais de endereço, os sinais de dados e os sinais de controle. A cada um destes grupos é associada uma linha de comunicação (ou um barramento), respectivamente, o barramento de endereços, o barramento de dados e o barramento de controle. Estes barramentos permitirão conectar os pinos do microprocessador aos pinos de mesma função dos circuitos de memória ou de E/S.

Um exemplo de funcionamento é o carregamento de uma instrução. O microprocessador inicialmente carrega o endereço da instrução no barramento de endereços; em seguida, ele ativa um sinal no barramento de controle para especificar à memória uma operação de leitura; em resposta, a memória vai colocar, no barramento de dados, a palavra representando a instrução requisitada e previne o microprocessador (com um sinal no barramento de controle); ao perceber o sinal de validação, o microprocessador lê a palavra no barramento de dados e a armazena num de seus registros internos. Em alguns casos, será necessária a leitura de outras palavras no barramento de dados para que a instrução possa ser executada. Nestes, casos, o processo descrito acima se repetirá pelo número de vezes correspondente ao número de palavras a serem lidas.

O valor do sinal elétrico num pino de um microprocessador (ou linha do barramento) indica se o sinal está ativo ou não. Alguns sinais são ativos no nível alto (ou seja, em 5 v), enquanto outros são ativos em nível baixo (ou seja, 0 v).

Dois parâmetros podem ser associados à performance de um microprocessador: o número de linhas do barramento de endereços e o número de linhas do barramento de dados.

Um microprocessador dotado de *m* linhas de endereço pode endereçar até 2^m palavras de memória. Os valores correntes de *m* são 16, 20, 24 ou 32. Da mesma forma, um microprocessador que dispõe de *n* linhas de dados pode ler ou escrever uma palavra de *n* bits numa única operação. Os valores correntes de *n* são 8, 16, 32 ou 64.

Um microprocessador que tenha um barramento de dados de 8 bits vai necessitar de quatro operações de leitura para acessar uma palavra de 32 bits.

Um microprocessador de 32 bits vai obter a mesma informação numa única operação e ele será, naturalmente mais rápido. Além das linhas de dados e de endereços, um microprocessador dispõe de linhas de controle. Estas permitem garantir a sincronização, a regulação e o controle das trocas de dados no barramento entre o

microprocessador e o seu ambiente. Outros sinais são ainda encontrados disponíveis, como a alimentação (5 V), o relógio e o terra.

Embora cada microprocessador terá definido seus sinais específicos, é possível, de uma maneira mais geral, classificá-los em seis categorias: controle do barramento, interrupções, controle de acesso ao barramento, estados funcionais, etc.

3. COMPONENTES DE UM MICROPROCESSADOR

Nesta seção, apresentaremos as principais características dos componentes da arquitetura de um microprocessador, em particular, os registros, os barramentos, a ALU, os multiplexadores e decodificadores, o deslocador, os relógios, e a memória principal.

3.1. Registros

Um registro é um dispositivo capaz de memorizar uma informação. Na arquitetura de um microprocessador, os registros, geralmente numerosos, são utilizados para assegurar o armazenamento temporário de informações importantes para o processamento de uma dada instrução.

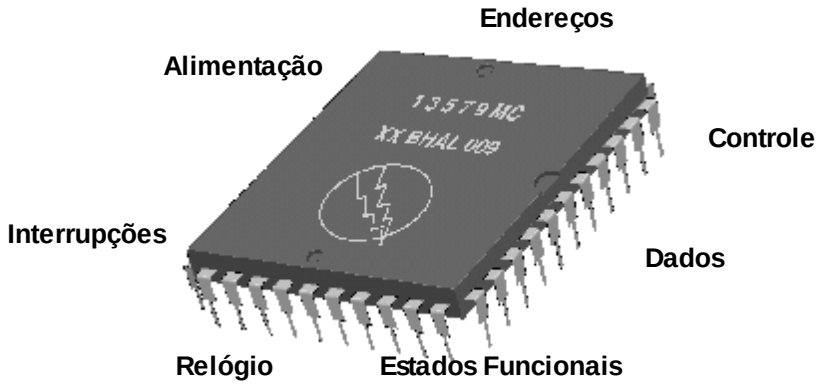

Figura 3.1 - Configuração externa de um microprocessador.

Conceitualmente, registro e memória são semelhantes... são a localização, a capacidade de armazenamento e os tempos de acesso às informações que os diferenciam. Os registros se localizam no interior de um microprocessador, enquanto a memória é externa a este. Um registro memoriza um número limitado de bits, geralmente uma palavra de memória.

Quanto mais complexos e mais potentes os microprocessadores, mais registros eles contém. No caso de pequenos computadores e pouco potentes, a memória principal é utilizada para armazenar valores intermediários em função do número reduzido de registros.

Os registros de um microprocessador constituem uma memória local, privada ou uma memória "bloco de notas" para o microprocessador. A memória central é considerada uma memória global compartilhada.

3.2. Barramentos

Um barramento é constituído de um conjunto de fios sobre os quais vão transitar sinais em paralelo. Os barramentos internos de um microprocessador são bem mais simples que os barramentos constituindo a arquitetura de um computador, pois eles conectam, em 99% dos casos unicamente dois componentes em modo ponto a ponto, o que significa que não existe a necessidade de se definir linhas de endereço e o número de linhas de comando é bastante reduzido. A transmissão de informação nos barramentos internos é feita em paralelo para aumentar a eficácia do processamento.

Um barramento pode ser unidirecional (a informação transita num único sentido) ou bidirecional (a informação vai transitar nos dois sentidos, de maneira alternativa, é claro!). Geralmente, os componentes conectados aos barramentos podem se conectar e se desconectar eletricamente. Estas conexões-desconexões são feitas num tempo muito curto (da ordem de nanosegundos). Um barramento que apresente estas propriedades é chamado um barramento de três estados, ou seja, cada uma de suas linhas pode assumir os valores lógicos 0, 1 e *flutuante*. Os

barramentos a três estados podem ser utilizados quando se necessita conectar diversos componentes que poderiam trocar dados através do barramento.

Em geral, os registros são conectados via barramentos conectando suas entradas e suas saídas. Um exemplo disto é mostrado na figura 3.2.

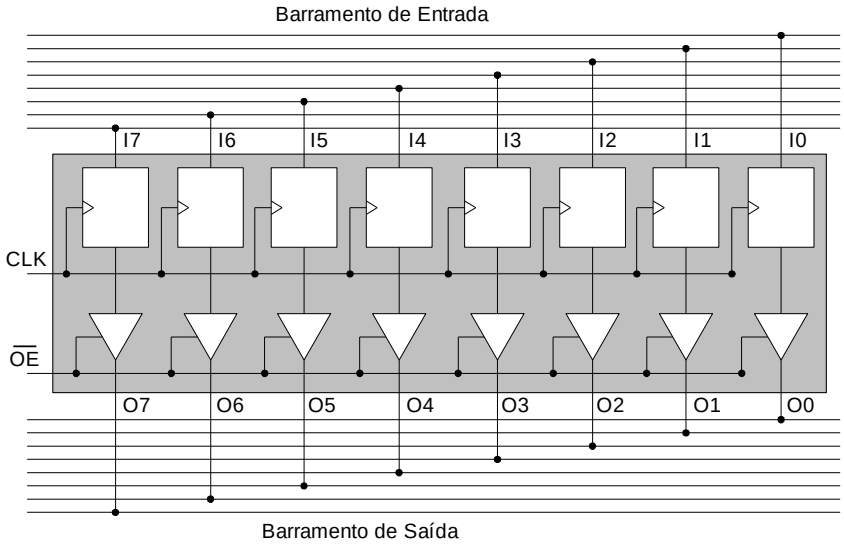

Figura 3.2 - Um registro de 8 bits conectado a dois barramentos (entrada e saída).

O registro é composto de 8 flip-flops do tipo D. Suas saídas são conectadas através de circuitos a três estados. O registro dispõe de dois sinais de comando: o sinal de relógio CK permite realizar o carregamento do registro; o sinal OE permite ativar os sinais de saída, colocando o conteúdo do registrador à disposição no barramento de saída.

Normalmente, os sinais CK e OE estão em repouso (inativos). Com CK inativo, o conteúdo do registro não é afetado pelas informações circulando no barramento de entrada.

Quando CK é ativado, o registro vai armazenar a informação presente no barramento de entrada, que vai compor então o conteúdo do registrador. Com OE em repouso, o registro é considerado "desconectado eletricamente" do barramento de saída. Quando OE é ativado, o conteúdo do registro será colocado no barramento de saída.

Desta maneira, se as saídas de um registro R1 estiverem conectadas às entradas de um registro R2, é possível copiar o conteúdo de R1 em R2. O sinal OE de R1 é ativado e mantido assim até a ativação do sinal CK de R2. Este tipo de operação é muito frequente em programação.

3.3. Unidade Lógica e Aritmética (ALU)

Os computadores utilizam diversos componentes específicos para efetuar os cálculos ou o tratamento de informação. O circuito de cálculo mais simples é o somador, que adiciona dois números de n bits. A ALU é o circuito que efetua diversas operações aritméticas e lógicas entre dois operandos.

O tipo de tratamento a efetuar deve ser informado através de sinais de seleção de operação. A figura 3.3 mostra a composição de uma ALU, capaz de realizar quatro diferentes operações com dois números binários A e B: A e B, A ou B, Complemento de B ($\overline{B}$) e A+B, a escolha das operações sendo feitas segundo o valor dos dígitos S0 e S1.

3.4. Multiplexadores e Decodificadores

O multiplexador apresenta 2^n entradas e uma única saída. Um grupo de linhas de seleção permite selecionar uma das 2^n entradas (uma simples linha ou um barramento) para ser dirigida à saída.

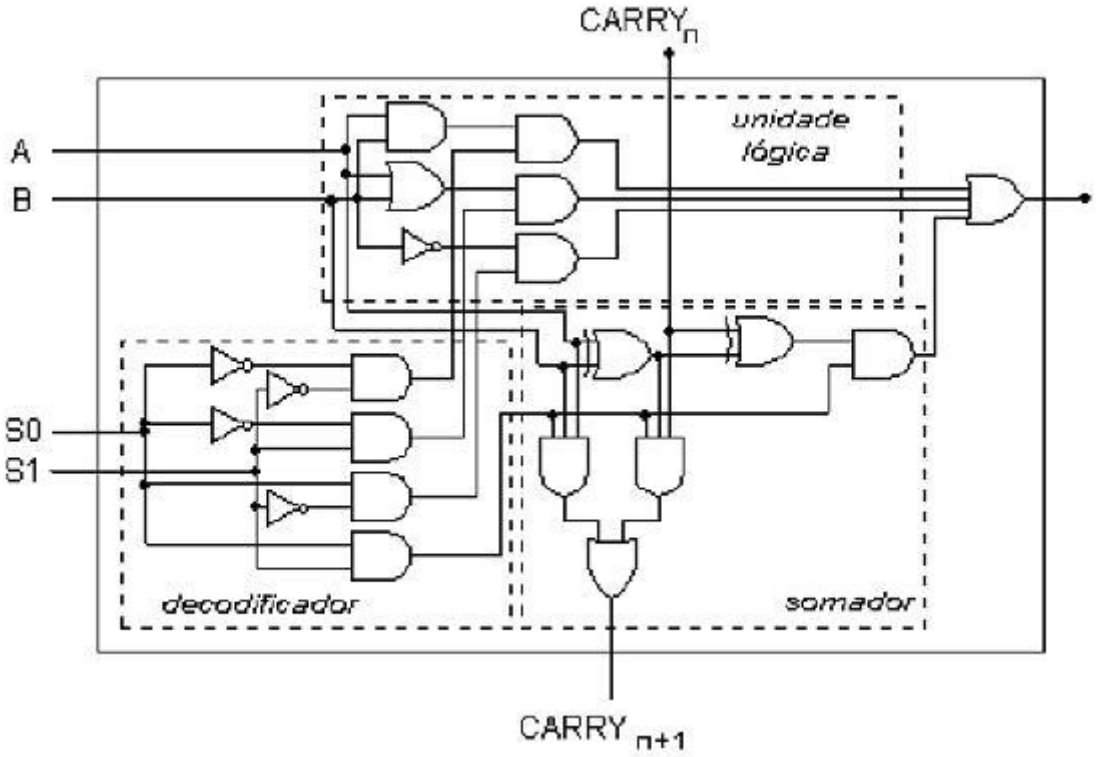

Figura 3.3 - ALU de quatro operações.

Nos esquemas de microprocessadores, os multiplexadores são simbolizados por MUX. A figura 3.4 apresenta um multiplexador de 2 entradas e 1 saída e um decodificador de 4 entradas por 16 saídas. Um circuito complementar, o demultiplexador realiza a função inversa do multiplexador, ou seja, sua função corresponde a dirigir a única entrada a uma das 2^n saídas, em função dos valores binários das linhas de seleção.

O decodificador compreende n entradas e 2^n saídas, ordenadas de 0 a 2^{n-1} . Se o valor presente sobre as entradas é k, apenas a saída de ordem k será ativa. O decodificador analisa a informação nas suas entradas e fornece na saída, de maneira exclusiva, a indicação ou significado desta informação dentre várias possibilidades.

3.5. Relógio

A maior parte dos circuitos compondo a arquitetura de um computador são circuitos lógicos síncronos, controlados por um relógio. O relógio é um circuito capaz de transmitir regularmente, segundo uma periodicidade determinada, pulsos elétricos, que permitem definir ciclos de máquina.

O intervalo de tempo entre dois pulsos é chamado período do relógio ou tempo de ciclo. As frequências de relógio estão geralmente entre 1 e 3000 MHz. Cada ciclo de máquina determina uma atividade de base no computador, como por exemplo, a busca de uma instrução, a execução de uma instrução, etc...

Numa CPU, uma atividade de base é variada e na maior parte das vezes complexa; algumas vezes, é necessário que um ciclo de relógio seja decomposto em vários sub-ciclos de maneira a organizar e sincronizar as ações do ciclo principal.

A figura 3.5 mostra um circuito de relógio que fornece quatro sinais na saída. O primeiro sinal (CK0) corresponde à saída principal, ou seja, o relógio de referência; os demais sinais (CK1, CK2 e CK3) são derivados do primeiro, cada um deles sofrendo um atraso diferente. A largura de pulso do sinal de referência é igual a 1/4 do ciclo de relógio. Cada um dos sinais sofre um atraso, respectivamente de uma, duas e três vezes a largura de pulso do sinal. Desta forma, é possível obter um circuito que divide um ciclo de relógio em quatro ciclos de mesma largura, defasados de uma largura de pulso.

4. CONCEITO DE MICROARQUITETURA

Uma vez revisto o funcionamento dos diferentes componentes de um microprocessador, é importante estudar como estes podem ser combinados de modo a obter um circuito capaz de interpretar e executar um dado conjunto de instruções de programa.

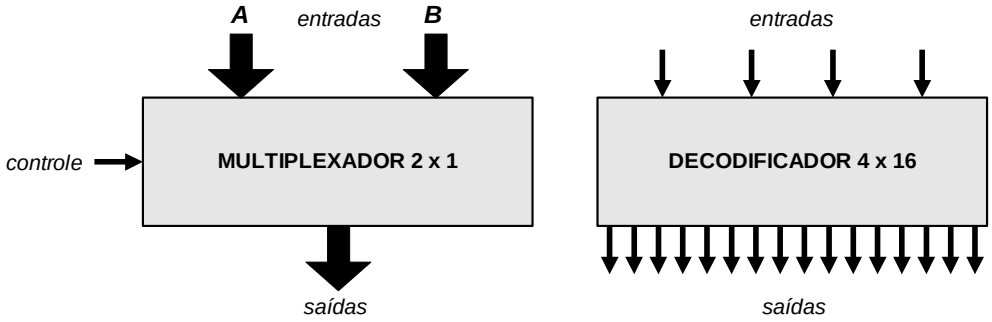

Figura 3.4 - Multiplexador 2 x 1 e Decodificador 4 x 16.

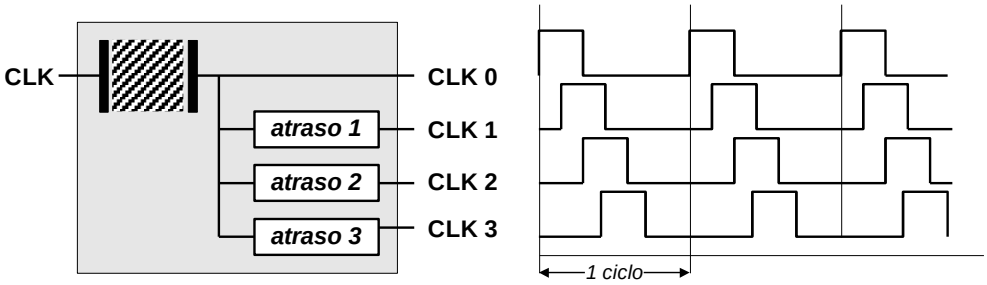

Figura 3.5 - Circuito de relógio e seu cronograma.

A arquitetura interna do microprocessador vai estabelecer o primeiro nível de linguagem que será implementada pelos circuitos constituintes.

4.1. Um exemplo de micromáquina

A figura 3.6 apresenta um exemplo de arquitetura interna de um microprocessador e a maneira como a informação será conduzida. Nas seções a seguir será feita a descrição dos seus diferentes elementos e os sinais utilizados para conduzi-los.

Na micromáquina apresentada, o caminho dos dados é composto de 16 registros gerais de 16 bits (A, B, C, PC, SP, AC, etc.), uma ALU e diversos barramentos (A, B e C). Cada registro pode transmitir seus dados aos barramentos A ou B e pode receber as informações sobre o barramento C.

Os barramentos A e B vão alimentar a ALU com operandos de 16 bits para realizar uma das quatro operações: A+B, A e B, A e complemento de A ($\overline{A}$). A seleção da operação a realizar será indicada pelos bits F0 e F1. Dois outros bits, N e Z, indicam, respectivamente, se o resultado da operação é negativo ou zero (estes sinais são ativados após uma operação da ALU).

A saída da ALU é conectada à entrada de um registrador de deslocamento, DESLOC, que efetua sobre o operando ou uma simples transferência, ou um deslocamento para a direita ou para esquerda, os bits utilizados para indicar a maneira de transmitir o dado sendo S0 e S1.

Os barramentos A e B não são conectados diretamente às entradas da ALU, servindo-se para isto de buffers A e B (manter estabilidade dos dados na entrada).

A troca de dados com a memória principal do microprocessador é feita graças a dois outros registros MAR(*Memory address Register*) e MBR(*Memory Buffer Register*). MAR pode ser carregado com um dado do barramento B, o carregamento deste registro sendo comandado pelo sinal M0.

Um dado na saída do registro de deslocamento pode ou ser carregado num dos 16 registros gerais, ou ser enviado à memória principal via registro MBR, o sinal utilizado para comandar o carregamento do registro sendo M1.

M2 e M3 permitem comandar, respectivamente as operações de escrita e leitura em memória. Um dado lido na memória pode ser carregado diretamente como entrada da ALU, segundo a seleção efetuada pelo multiplexador AMUX, A0 sendo o sinal utilizado para o controle.

MICROINSTRUÇÕES

O controle do caminho dos dados do microprocessador vai necessitar então de 61 sinais:

- 16 para comandar a descarga dos registros sobre o barramento A
- 16 para comandar a descarga dos registros sobre o barramento B
- 16 para comandar a carga dos registros a partir do barramento C
- 2 sinais de carregamento dos buffers A e B (L0 e L1)
- 2 sinais para definir a operação a executar (F0 e F1)
- 2 sinais para o comando do registrador de deslocamento (S0 e S1)
- 4 sinais para comandar os registros MAR e MBR (M0, M1, M2 e M3)
- 2 sinais para indicar a operação de leitura ou escrita com a memória principal
- 1 sinal para controlar o multiplexador

A micromáquina disporia então de um registro de comando de 61 bits que permitirá controlar, através dos valores dos bits, as operações a serem efetuadas pelo microprocessador.

Pode-se ainda reduzir o número de bits necessários para definir uma instrução, com um pequeno aumento no hardware interno da máquina.

Com o uso de decodificadores para endereçar os 16 registros em cada um dos barramentos (A, B e C) o número de bits necessários para este fim cai de 48 para 12, o que reduz o número total de bits de 61 para 25. L0 e L1 podem, de fato utilizar sinais já fornecidos pelo relógio, uma vez que estes serão ativados em ciclos bem precisos da execução_logo, são menos dois sinais (23).

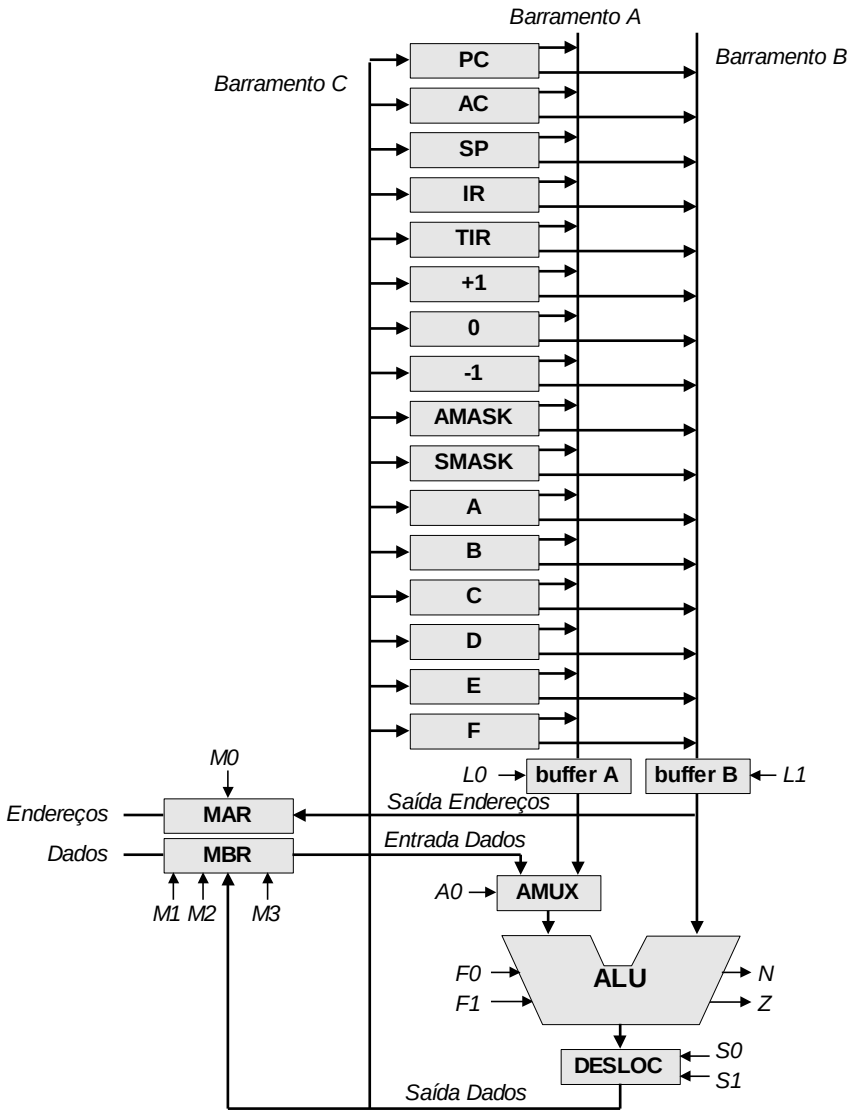

Figura 3.6 - Exemplo de microarquitetura.

O número de bits, por outro lado, é aumentado se considerarmos que vamos necessitar de um sinal especial para autorizar ou não o carregamento de um registro sobre C com o resultado de uma operação (sabendo que algumas vezes a operação vai ser executada simplesmente para ativar os sinais N e Z)... 24 sinais serão necessários. M2 e M3 podem ser comandados por RD e WR... assim, uma microinstrução do microprocessador poderá ser composta de 22 bits.

4.2. Formato das microinstruções

A figura 3.7 apresenta o formato das microinstruções da micromáquina definida na figura 3.6. Como pode ser visto, ela é composta de diversas partes, chamadas campo, onde cada campo vai ter um significado especial para a operação da micromáquina. Podemos observar ali 13 campos, dos quais onze são definidos a seguir:

AMUX (1 bit) - controla a entrada esquerda da ALU: 0=*latch* A, 1 = MBR
ALU (2 bits) - função da ALU: 0=A+B, 1=A AND B, 2=A, 3=A' (complemento)
DESLOC (2 bits) - função do deslocador:0=nenhum desloc., 1= à direita 2= à esquerda
MBR (1 bit) - carrega MBR a partir do deslocador: 0= não carrega, 1 = carrega MBR
MAR(1 bit) - carrega de MAR a partir do *latch* B: 0=não carrega, 1= carrega MAR
RD (1 bit) - leitura da memória: 0=nenhuma leitura, 1= carrega MBR da memória
WR (1 bit) - escrita na memória: 0=nenhuma escrita, 1= escreve MBR na memória

VALC (1 bit) - autorização de carregamento dos registros: 0=não carrega, 1=carrega
C (4 bits) - endereçamento dos registros para o barramento C se VALC=1
B (4 bits) - endereçamento dos registros para o barramento B
A (4 bits) - endereçamento dos registros para o barramento A

Os campos **COND** e **END** serão definidos mais adiante.

4.3. Ciclo de execução de uma instrução

A execução de uma microinstrução pelos diversos componentes da micromáquina deve obedecer a um ciclo bem definido. O ciclo de execução de uma instrução pela ALU é composto das seguintes etapas:

- carregamento dos operandos nos buffers A e B
- fornecimento de tempo suficiente para a ALU e o registro de deslocamento para realizar suas funções
- carregamento do resultado da operação num registro geral ou no registro MBR.

É evidente que estas etapas devem ser realizadas de maneira sequencial e não simultaneamente, o que impõe uma certa ordenação no tempo. Isto significa que é preciso dispor de um circuito de relógio que gere diferentes sinais (4 subciclos, por exemplo) que vão definir as diferentes fases da execução de uma microinstrução.

As ações que seriam executadas nos quatro sub-ciclos seriam as seguintes:

❶ Carregamento da microinstrução num registro particular, chamado registro de microinstrução (RMI);
❷ Transferência do conteúdo dos registros selecionados sobre os barramentos A e B, e após carregamento dos dados correspondentes nos buffers A e B;
❸ Execução da operação pela ALU;
❹ Terminada a operação, o dado de saída (resultado da operação) vai ser carregado, ou num registro geral ou no registro MBR.

A figura 3.8 apresenta um esquema completo da micromáquina, sobre o qual podemos distinguir dois subconjuntos de circuitos: à esquerda o caminho dos dados que foi analisado anteriormente; à direita, uma unidade de comando que será analisada a seguir. A memória de controle é a parte mais importante desta unidade pois é ela que vai conter as microinstruções da micromáquina. Na maioria das máquinas, esta memória é uma ROM.

No exemplo considerado, as microinstruções são codificadas em palavras de 32 bits. A memória de comando selecionada, vai ter a capacidade de armazenar até 256 palavras de 32 bits (256 x 32 bits).

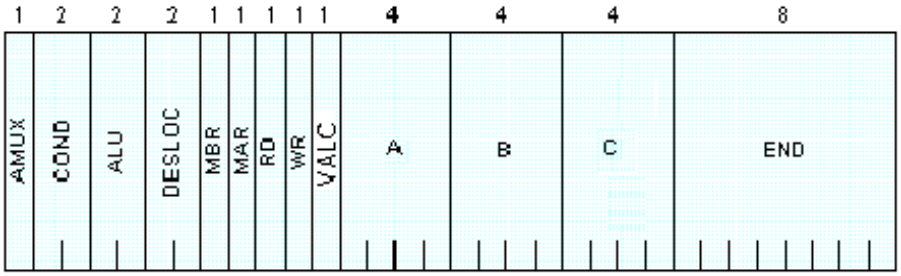

Figura 3.7 - Formato de uma microinstrução da micromáquina exemplo.

A memória de comando dispõe de um registro de endereços (MPC) e de um registro de dados (RMI). O primeiro registro permite endereçar a microinstrução a ser executada e o segundo armazená-la.

A memória de comando atua no primeiro subciclo da execução de uma instrução, fazendo a carga da microinstrução apontada por MPC no registro RMI. No subciclo 2 o conteúdo de RMI é estável, sendo que as informações contidas nos campos da microinstrução vão agir no caminho de dados, particularmente os campos A e B que vão determinar quais as informações deverão ser carregadas nos barramentos A e B (sendo que os decodificadores vão realizar a decodificação , suas saídas agindo sobre as entradas OE dos registros selecionados).

Ainda no subciclo 2, o relógio permite carregar as informações sobre os barramentos A e B nos buffers, respectivamente A e B. Assim, para os subciclos seguintes, os dados de entrada da ALU estarão estáveis nestes buffers.

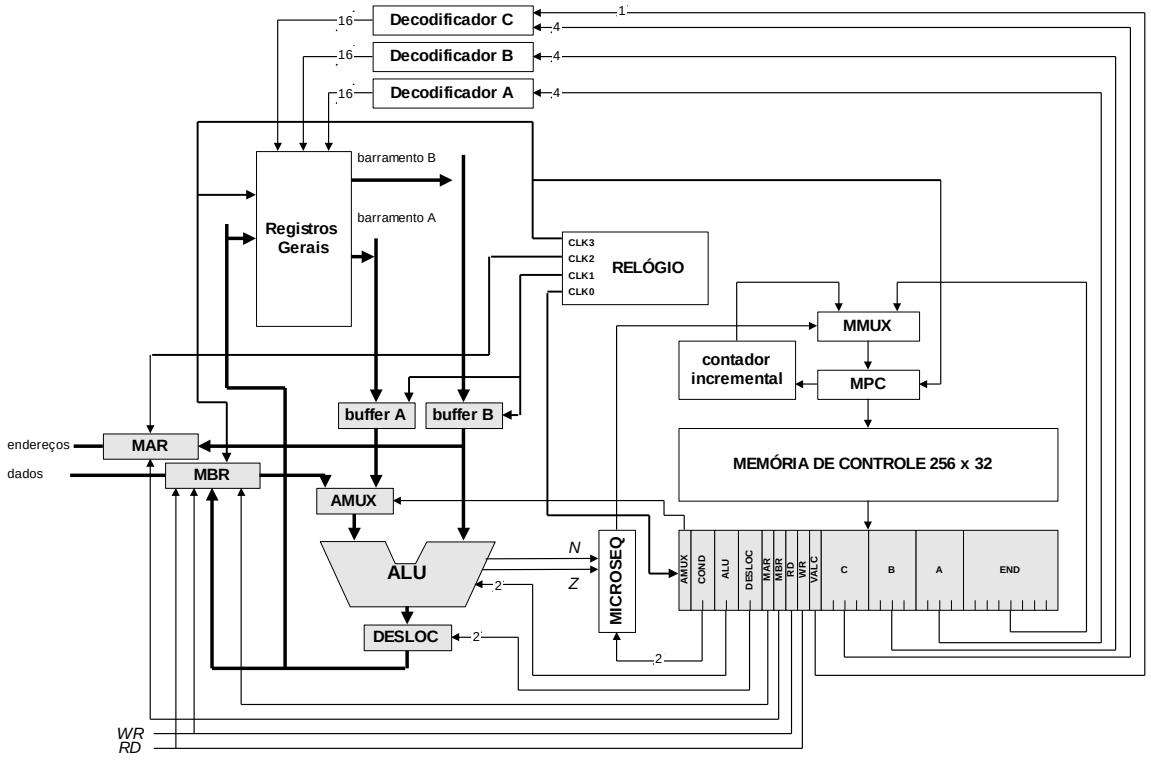

Figura 3.8 - Micromáquina completa.

Durante esta operação, o registro MPC é incrementado de um por um circuito especial — o contador incremental, o que permite preparar o carregamento da próxima microinstrução no RMI, aumentando assim a velocidade de execução do microprograma.

No subciclo 3, a ALU e o registro de deslocamento vão executar as operações sobre os dados, segundo os valores dos campos ALU e DESLOC. Além disso, o campo AMUX que comanda o multiplexador AMUX vai permitir indicar se o dado a ser carregado na entrada esquerda da ALU deve ser o conteúdo do buffer A ou o conteúdo do registro MBR.

Por outro lado, o buffer B estará sempre conectado à entrada direita da ALU. O tempo de cálculo da ALU pode variar segundo a operação a efetuar. No nosso caso, a operação mais longa é a de adição. De maneira paralela aos tratamentos efetuados pela ALU e pelo registro de deslocamento. O registro MAR pode ser carregado com o valor contido no buffer B se o campo MAR indicar assim. Durante o subciclo 4 (o último), o dado que se encontra sobre o barramento C vai ser carregado num dos 16 registros gerais ou no registro MBR, segundo as indicações dos campos VALC e MBR da microinstrução.

O registro MBR será também carregado no subciclo 4 se o campo MBR indicar. Os sinais de comando RD e WR são difundidos para a memória principal e para o registro MBR, durante todo o tempo em que eles estejam estáveis no registro RMI.

4.4. Sequenciamento das microinstruções

Acabamos de analisar a execução de uma única microinstrução. Vejamos agora como um conjunto de microinstruções é executado. Na maioria dos casos, as microinstruções são executadas de modo sequencial, ou seja, a microinstrução encontra-se no endereço que sucede o da que está sendo processada. Em certos casos, portanto, é necessário romper o sequenciamento normal das microinstruções, pela introdução de ordens de quebra deste

sequenciamento. Isto é feito graças aos dois campos adicionais da microinstrução (que não haviam sido definidos até então) que são os campos END e COND.

O campo END (8 bits) permite (em caso de quebra do sequenciamento normal de execução) indicar o endereço da eventual microinstrução a executar. O campo COND vai permitir indicar qual será a próxima instrução a executar: ou a instrução em MPC + 1 ou a instrução em END. Desta forma, cada microinstrução vai conter a possibilidade de um salto condicional (ou não).

Em nossa micromáquina, a escolha da microinstrução a executar é definido por um circuito especial (o micro sequenciador) durante o subciclo 4 uma vez que os bits N e Z estão estáveis na saída da ALU. A saída deste micro sequenciador é conectada à entrada de um multiplexador MMUX, que vai carregar o registro MPC seja com MPC + 1, seja com END, segundo o que estiver definido pelo campo COND:

COND = 0 sem quebra de sequenciamento --- MPC = MPC + 1
COND = 1 quebra de sequenciamento se N = 1
COND = 2 quebra de sequenciamento se Z = 1
COND = 3 quebra de sequenciamento incondicional

O micro sequenciador implementa a seguinte equação booleana, levando em conta as saídas da ALU, N e Z e o campo COND, que é aqui representado pelos bits L (da esquerda) e R (da direita):

$$MMUX = \bar{L}\ R\,N + L\ \bar{R}\ Z + L\,R = RN + LZ + LR$$

Assim MMUX será igual a 1 nas seguintes condições:

• $LR = 01_2$ *e* $N = 1$ • $LR = 10_2$ *e* $Z = 1$ • $LR = 11_2$

4.5. **Exemplos de microinstrução**

Alguns exemplos de microinstrução comuns na nossa microarquitetura são apresentadas na figura 3.9.

AMUX	COND	ALU	DESL	MBR	MAR	RD	WR	ENC	C	B	A	END	OPERAÇÃO
0	0	0	0	0	1	1	0	0	0	0	0	00	*mar:=pc; rd;*
0	0	2	0	0	0	1	0	0	0	0	0	00	*rd;*
0	0	0	0	0	0	0	0	1	0	6	0	00	*pc:=pc+1;*
0	1	2	0	0	0	0	0	0	0	0	4	15	*alu:=tir;* ***if*** *n* ***then goto*** *15*
0	1	0	2	0	0	0	0	1	4	3	3	69	*tir:=lshift(ir+ir);* ***if*** *n* ***then goto 69****;*

Figura 3.9 - Exemplos de microinstruções.

5. LINGUAGEM DE MÁQUINA E MICROCÓDIGO

5.1. Características básicas do microprocessador

O programador do microprocessador considerado vê o componente como uma máquina composta de 4 Kb de memória e de três registros específicos: o contador de programa (PC), o acumulador (AC) e o apontador de pilha (SP).

5.2. Modos de endereçamento de dados

As referências aos dados poderão ser feitos segundo três diferentes maneiras:

- o modo direto, onde as instruções serão compostas de um endereço absoluto de 12 bits, correspondente à posição do dado na memória; é o modo de endereçamento mais indicado para fazer acesso a variáveis globais do programa;
- o modo indireto, onde o endereço é obtido por cálculo, sendo o resultado do cálculo armazenado no registro AC; uma aplicação típica deste modo é o acesso a componentes de vetores ou tabelas em programas que utilizem tais estruturas;
- o modo local, que permite localizar uma informação na área de pilha do programa, com relação ao endereço apontado pelo registro SP; com este modo de endereçamento, é possível realizar acesso a dados presentes na pilha, particularmente, parâmetros de rotinas.

5.3. Conjunto de Instruções da Microarquitetura

A linguagem de máquina da microarquitetura considerada é composta de 23 instruções. A tabela a seguir apresenta cada uma das instruções da linguagem, juntamente com uma breve descrição de sua operação.

Código	Descrição
LODD end	Carrega em *AC* o dado posicionado na posição indicada por *end*
STOD end	Armazena o conteúdo do acumulador no endereço indicado por *end*
ADDD end	Adiciona o dado posicionado em *end* ao conteúdo do acumulador
SUBD end	Subtrai o dado posicionado em *end* do conteúdo do acumulador
JPOS end	Salto condicional para *end* se conteúdo do acumulador é positivo
JZER end	Salto condicional para *end* se conteúdo do acumulador é zero
JUMP end	Salto incondicional para *end*
LOCO dado	Carrega acumulador com valor indicado por *dado*
LODL x	Carrega o acumulador com dado posicionado a *x* posições do *SP*
STOL x	Armazena o conteúdo do acumulador na pilha a *SP + x*
ADDL x	Adiciona o conteúdo do acumulador com dado armazenado em *SP + x*
SUBL x	Subtrai o dado armazenado em *SP + x* do conteúdo do acumulador
JNEG end	Salto condicional para *end* se conteúdo do AC é negativo
JNZE end	Salto condicional para *end* se conteúdo do AC é diferente de zero
CALL end	Chamada de subrotina localizada em *end*
PSHI	Transfere para a pilha o conteúdo de memória apontado pelo *AC*
POPI	Transfere o conteúdo da pilha para a posição apontada por *AC*
PUSH	Transfere para a pilha o conteúdo do registro *AC*
POP	Transfere o conteúdo da pilha para o registro *AC*
RETN	Retorno de subrotina
SWAP	Troca os conteúdos do *AC* e da pilha
INSP y	Incrementa o *SP* de *y* unidades
DESP y	Decrementa o *SP* de *y* unidades

Na descrição dos códigos das instruções na tabela, os operandos apresentados correspondem ao seguinte:

- *end* são endereços expressos em 12 bits;
- *dado* são valores expressos em 12 bits;
- *x* são deslocamentos expressos em 12 bits;
- *y* são deslocamentos expressos em 8 bits.

Para um melhor entendimento do significado das instruções da linguagem, são apresentados, na próxima tabela, os códigos em binário e as operações realizadas sobre os registros da microarquitetura.

Código	Binário	Operação
LODD end	`0000 xxxx xxxx xxxx`	`ac = m[x]`
STOD end	`0001 xxxx xxxx xxxx`	`m[x] = ac`
ADDD end	`0010 xxxx xxxx xxxx`	`ac = ac + m[x]`

SUBD end	0011 xxxx xxxx xxxx	ac = ac - m[x]
JPOS end	0100 xxxx xxxx xxxx	if ac > 0 then pc = x
JZER end	0101 xxxx xxxx xxxx	if ac = 0 then pc = x
JUMP end	0110 xxxx xxxx xxxx	pc = x
LOCO dado	0111 xxxx xxxx xxxx	ac = x , 0 ≤ x ≤ 4095
LODL x	1000 xxxx xxxx xxxx	ac = m[sp+x]
STOL x	1001 xxxx xxxx xxxx	m[sp+x] = ac
ADDL x	1010 xxxx xxxx xxxx	ac = ac + m[sp+x]
SUBL x	1011 xxxx xxxx xxxx	ac = ac - m[sp+x]
JNEG end	1100 xxxx xxxx xxxx	if ac < 0 then pc = x
JNZE end	1101 xxxx xxxx xxxx	if ac ≠ 0 then pc = x
CALL end	1110 xxxx xxxx xxxx	sp = sp - 1; m[sp] = pc; pc = x
PSHI	1111 0000 0000 0000	sp = sp - 1; m[sp] = m[ac]
POPI	1111 0010 0000 0000	m[ac] = m[sp]; sp = sp + 1;
PUSH	1111 0100 0000 0000	sp = sp - 1; m[sp] = ac
POP	1111 0110 0000 0000	ac = m[sp]; sp = sp + 1;
RETN	1111 1000 0000 0000	pc = m[sp]; sp = sp + 1
SWAP	1111 1010 0000 0000	tmp = ac; ac = sp; sp = tmp
INSP y	1111 1100 yyyy yyyy	sp = sp + y , 0 ≤ x ≤ 255
DESP y	1111 1110 yyyy yyyy	sp = sp - y , 0 ≤ x ≤ 255

5.4. O Microprograma

Um aspecto interessante a ser destacado aqui é o que define a linguagem de máquina da microarquitetura considerada. Um fator determinante para a especificação da linguagem de máquina é sem dúvida a própria microarquitetura, pois esta vai limitar as operações que poderão ser realizadas em função de seus componentes.

Por outro lado, o que dará à máquina o poder de interpretar e executar as instruções de máquina definidas na linguagem é o microprograma que estará armazenado na memória de controle. O microprograma atua como um interpretador que controla todos os componentes da microarquitetura no sentido de realizar todos os passos da execução de um programa em linguagem de máquina. As operações básicas realizadas pelo interpretador são:

- a busca do código da instrução a executar;
- a decodificação da instrução;
- a execução propriamente dita da instrução.

Como já foi discutido, esta sequência de passos é executada num ciclo eterno, o que permite dar ao microprocessador a capacidade de executar seus programas em binário, instrução por instrução como estabelece o fluxo de controle de cada programa.

Nas seções a seguir, será explicada, através de exemplos, como são implementados, no interpretador, cada um dos passos citados acima.

5.5. O ciclo de busca de uma instrução

O ciclo de busca de uma instrução envolve um pequeno conjunto de registros da microarquitetura, particularmente, o PC, o MAR e o IR. Os passos básicos deste ciclo são:

❶ Carregamento do conteúdo do registro PC no registro MAR;
❷ Incremento do registro PC;
❸ Cópia do conteúdo da posição de memória apontada por MAR (código da instrução a executar) no registro IR;

Traduzindo estes passos para a sintaxe da linguagem de microprogramação, pode-se escrever o seguinte:

```
0: mar := pc; rd;                       /  carregamento do MAR
1: pc := pc + 1; rd;                    /  incremento do PC
2: ir := mbr; if n then goto 28;        /  código no IR
```

5.6. A decodificação das instruções

Considerando que o código das instruções a executar (armazenado em IR) é binário e que cada instrução é composta de uma parte mais significativa da palavra que indica o tipo de operação a ser executada, a decodificação das instruções é feita através da realização de operações de deslocamento (para a esquerda) da palavra que representa este código. Dependendo do resultado do deslocamento, pode-se então encaminhar (através de comandos **go to**) o microprograma para a seção de microinstruções que implementa a instrução identificada.

Na realidade, a decodificação das instruções é realizada através de um conjunto sequencial de testes (if-then-else) onde a condição testada é a pertinência do código a um dado intervalo de valores (o que especifica o tipo de operação a realizar). Em caso positivo, o controle é transferido para uma região do microprograma onde o microcódigo relativo àquele grupo de instruções está implementado; caso contrário, o controle é transferido para uma outra região de teste do microprograma (para verificar a pertinência da instrução a um outro grupo de instruções).

No exemplo a seguir, vemos um trecho de microprograma capaz de interpretar duas diferentes instruções: a instrução de carregamento de um dado constante (**LOCO dado**) e uma instrução de rotação do conteúdo do acumulador para a esquerda (**ROL**). O código de máquina da instrução **LOCO dado** é **0111 xxxx xxxx xxxx**, onde os 12 bits a **x** identificam o valor do dado a ser carregado no acumulador. Em hexadecimal, considerando que o dado a carregar é **A30**, o código de máquina desta instrução fica **7A30**. A instrução **ROL** não foi definida anteriormente. Por isso, podemos definir seu código como sendo **E008**.

O microcódigo mostrado a seguir ilustra a situação de decodificação onde apenas uma das duas instruções pode estar presente no programa em linguagem de máquina. Na linha 4, é realizada uma operação de AND lógico entre o conteúdo do registro IR (o código da instrução) e o registro AMASK. Este registro da micromáquina contém uma constante cujo valor é **0000 1111 1111 1111** (ou **0FFF** em hexadecimal) cuja função é retirar, do código da instrução a executar, os 12 bits menos significativos. Continuando com a análise da linha 4, o objetivo da operação é verificar se o código sob análise é da operação de rotação ou da operação nula (0000 em hexadecimal). Nesta linha, está explícito que, se o resultado do AND lógico for zero, então o controle do programa deve ser transferido para a linha 0. Neste caso, está se considerando que o código pertence à instrução nula e não deve, então ser feito nada, retornando para ir buscar a próxima instrução. Caso contrário, o controle do microprograma é transferido para a linha seguinte que deve implementar a instrução de rotação.

```
3 :  ac:= band(ir,amask); goto 0;
4 :  a := band(ir, amask); if z then goto 0; /rotação esq.
```

5.7. A implementação das instruções

Vamos aproveitar o exemplo da instrução de rotação para ilustrar como as instruções de máquina são implementadas em microcódigo. A sequência de microinstruções a seguir será comentada no sentido de explicar esta implementação. Esta instrução de rotação à esquerda permite indicar, através do valor contido no registro A, de quantos bits será realizado o deslocamento.

```
5 :  ac  := lshift(ac);    if n then goto 7;
6 :  goto 8;
7 :  ac := ac + 1;
8 :  a   := a + -1; if z then goto 0;
9 :  goto 5;
```

Na linha 5, é realizada efetivamente o deslocamento para a esquerda do conteúdo do acumulador, expressa pela função **lshift(ac)**. No entanto, para que a rotação se caracterize (e não um deslocamento), é preciso garantir

que o bit mais significativo da palavra seja transferido para o bit menos significativo após a rotação, como mostra a figura 3.10.

Este é o papel das instruções que seguem a partir da linha 5. Na própria instrução 5, o resultado da rotação é verificado através do sinal N. Caso N seja 1, o que significa que o bit mais significativo da palavra é 1, o controle do microprograma é transferido para a linha 7, onde o conteúdo do registro AC é adicionado a 1. Caso contrário, o controle passa para a linha 6, que, por sua vez, provoca um salto para a linha 8. A partir da linha 8, é feito o controle do número de vezes que será feito o deslocamento. O controle é feito, decrementando-se o conteúdo do registro A e comparando o resultado do decremento com zero. Se o resultado for zero, não deverá haver mais rotação e a operação é considerada encerrada (**`if z then goto 0`**); se o resultado do decremento não for zero, o controle do microprograma é transferido mais uma vez para a linha 5 para que seja efetuada uma nova rotação de um bit para a esquerda (**`goto 5`**).

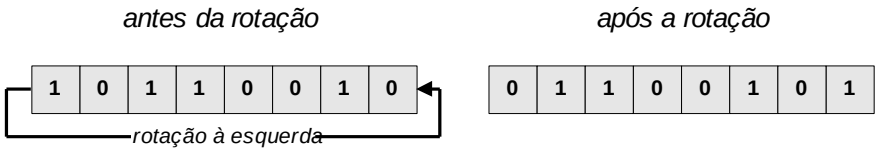

Figura 3.10 - Ilustração da operação de rotação à esquerda.

5.8. O microprograma completo

O microprograma capaz de interpretar e executar todas as instruções descritas na seção 4.3 é apresentado a seguir. Os comentários, separados por uma barra inclinada, auxiliam na compreensão do mesmo.

```
0: mar := pc; rd;                                  /  loop principal
1: pc := pc + 1; rd;                               /  incremento do PC
2: ir := mbr; if n then goto 28;                   /  salva e decodifica MBR
3: tir :=lshift(ir + ir); if n then goto 19;
4: tir :=lshift(tir); if n then goto 11;           /  000x ou 001x?
5: alu := tir; if n then goto 9;                   /  0000 ou 0001?
6: mar := ir; rd;                                  /  0000 = LODD
7: rd;
8: ac := mbr; goto 0;
9: mar := ir; mbr := ac; wr;                       /  0001 = STOD
10: wr; goto 0;
11: alu := tir; if n then goto 15;                 /  0010 ou 0011?
12: mar:= ir; rd;                                  /  0010 =ADDD
13: rd;
14: ac := mbr + ac; goto 0;
15: mar:= ir; rd;                                  /  0011 = SUBD
16: ac := ac + 1; rd;                              /  Obs: x-y = x+1 + not(y)
17: a := inv(mbr);
18: ac := ac + a; goto 0;
19: tir :=lshift(tir); if n then goto 25;          /  010x ou 011x?
20: alu :=tir; if n then goto 23;                  /  0100 ou 0101?
21: alu :=ac; if n then goto 0;                    /  0100 = JPOS
22: pc := band (ir, amask); goto 0;                /  realiza o salto
23: alu := ac; if z then goto 22;                  /  0101 = JZER
24: goto 0;                                        /  sem salto
25: alu:= tir; if n then goto 27;                  /  0110 ou 0111?

26: pc := band(ir, amask); goto 0;                 /  0110 = JUMP
27: ac :=band(ir, amask); goto 0;                  /  0111 = LOCO
28: tir :=lshift(ir + ir); if n then goto 40;      /  10xx ou 11xx?
29: tir :=lshift(tir); if n then goto 35;          /  100x ou 101x?
30: alu := tir; if n then goto 33;                 /  1000 ou 1001?
31: a :=ir + sp;                                   /  1000 = LODL
32: mar :=a; rd; goto 7;
33: a:=ir + sp;                                    /  1001 = STOL
34: mar := a; mbr:= ac;wr; goto 10;
35: alu := tir; if n then goto 38;                 /  010 ou 1011?
36: a :=ir + sp;                                   /  1010 = ADDL
37: mar := a;rd; goto 13;
38: a :=ir + sp;                                   /  1011 = SUBL
39: mar :=a;rd; goto 16;
40: tir :=lshift(tir); if n then goto 46;          /  110x ou 111x?
41: alu :=tir; if n then goto 44;                  /  1100 ou 1101?
42: alu := ac; if n then goto 22;                  /  1100 = JNEG
43: goto 0;
44: alu := ac; if z then goto 0;                   /  1101 = JNZE
45: pc := band(ir, amask); goto 0;
46: tir :=lshift(tir); if n then goto 50;
47: sp := sp + -1     ;                            /  1110 = CALL
48: mar := sp; mbr:= pc;wr;
49: pc := band(ir, amask);wr; goto 0;
50: tir := lshift(tir);if n then goto 65;          /  1111, busca endereço
51: tir := lshift(tir); if n then goto 59;
52: alu := tir; if n then goto 56;
53: mar :=ac;rd;                                   /  1111000 = PSHI
```

```
54: sp :=sp + -1     ;rd;
55: mar := sp;wr;goto 10;
56: mar := sp; sp:= 1 + sp;rd;                  /  1111001 = POPI
57: rd;
58: mar :=ac;wr;goto 10;
59: alu :=tir; if n then goto 62;
60: sp :=sp + -1     ;                          /  1111010 = PUSH
61: mar := sp;mbr := ac; wr;goto 10;
62: mar := sp;sp := 1 + sp;rd;                  /  1111011 = POP
63: rd;
64: ac:= mbr; goto 0;
65: tir := lshift(tir); if n then goto 73;
66: alu := tir; if n then goto 70;
67: mar := sp;sp := 1 + sp;rd;                  /  1111100 = RETN
68: rd;
69: pc := mbr; goto 0;
70: a := ac;
71: ac := sp;
72: sp := a; goto 0;
73: alu := tir; if n then goto 76;
74: a := band(ir, smask);                       /  1111110 = INSP
75: sp := sp + a; goto 0;
76: a:= band(ir, smask);                        /  1111111 = DESP
77: a:= inv(a);
78: a:= a + 1; goto 75;
```