

Proposta de arquitectura e organização de informação gráfica de um visualizador baseado num sistema Raster

Adolfo Steiger Garção — Eq. Prof. Aux. responsável pelo grupo de «Hardware» do Dep. de Informática da U.N.L. e
 António Maria Sarmento — Assistente do grupo «Hardware» do Dep. de Informática da U.N.L.

INTRODUÇÃO

Na utilização de sistemas Raster em visualizadores alfanuméricos e gráficos existem condicionamentos devido ao modo como a informação é descrita no écran e à maneira como deverá ser organizada a fim de permitir a sua manipulação interactiva (fig. 1).

Embora se trate da mesma informação a sua estrutura é completamente diferente. A conversão (e comunicação) entre elas poderá ser assegurada por uma interface simultaneamente apoiada em *hardware*, *firmware* e *software*. Esta interface faz parte integrante da arquitectura.

CARACTERÍSTICAS DO SISTEMA RASTER

A velocidade de afixação de pontos luminosos num sistema Raster é extremamente elevada. Num televisor normal podem obter-se pontos luminosos separados de cerca de 80 ns e com duração de cerca de metade deste valor. Há contudo que sujeitar a realização destes pontos sobre o écran ao processo próprio de descrição gráfica do sistema, ou seja saída por linha horizontal (cerca de 480 pontos são acessíveis) e um número de linhas que dependerá da utilização ou não de interlace e da zona do écran que se considere linear para os efeitos pretendidos (240 ou 480 constituem um compromisso neste sistema) (fig. 2).

Considerando a afixação de pontos luminosos sobre o écran como uma forma de tratamento da informação extremamente rápida, ter-se-á que propor um processador adequado a estas velocidades. Não surpreenderá a necessidade de realização de operações em paralelo e a relativa simplicidade das acções a serem executadas por uma parte deste processador.

Por outro lado ter-se-á que deixar para outra zona da arquitectura o tratamento da informação estruturada (listas) que permitam um processo interactivo. Este tratamento necessitará de meios computacionais (micro-computador) e será naturalmente muito mais lento.

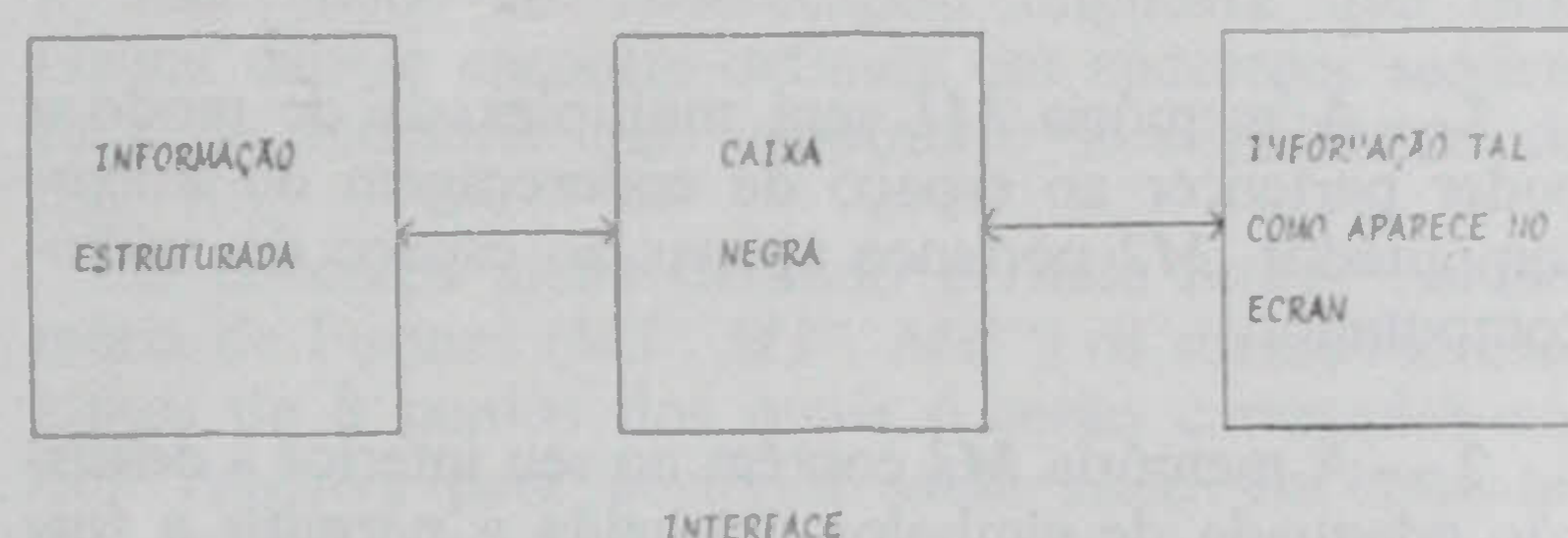


Fig. 1

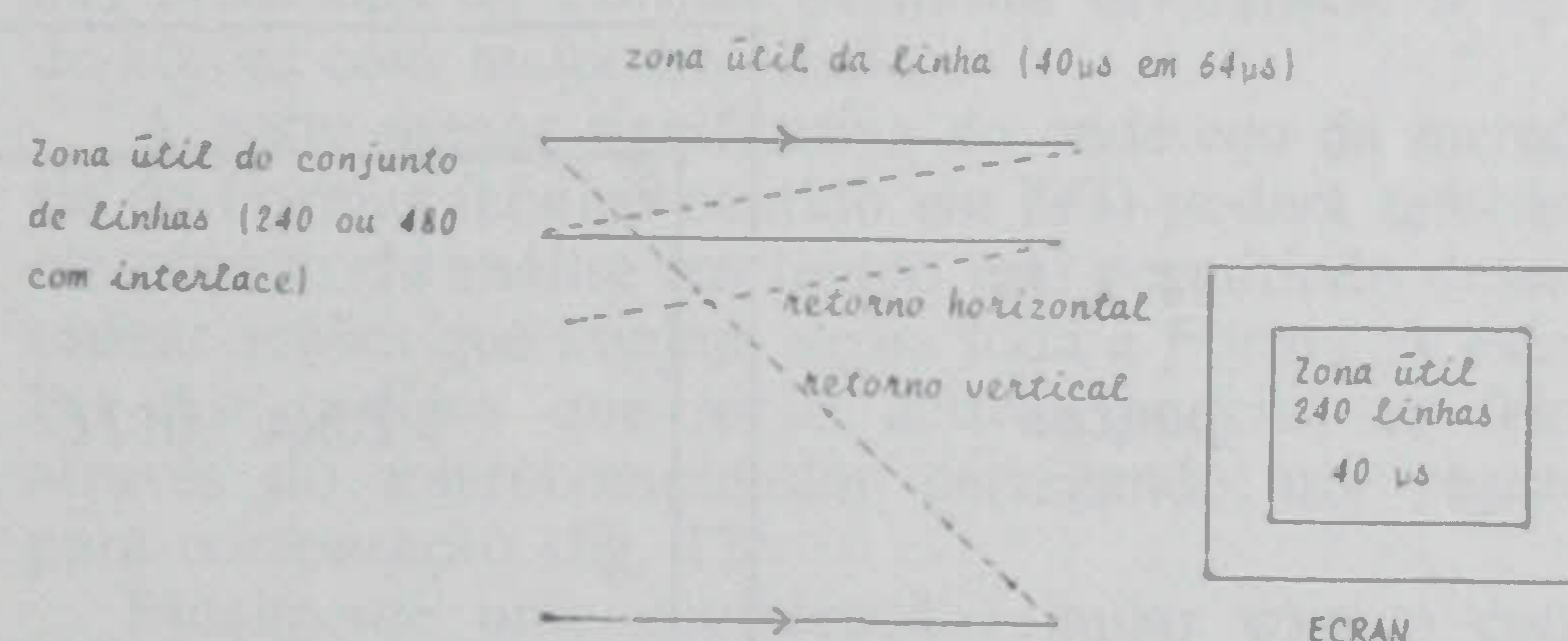


Fig. 2

PONTO DE PARTIDA DA ARQUITECTURA

Desenvolveu-se primeiramente uma arquitectura que permitisse materializar um visualizador alfanumérico no espaço adequado do écran (240×480 pontos) e que contivesse já os elementos necessários à sua adaptação a processador gráfico (fig. 3).

Pretendeu-se ainda obter uma solução económica, versátil e capaz de actuação autónoma.

Desta arquitectura reteremos os elementos fundamentais para a definição do processador gráfico admitindo que uma descrição mais detalhada poderá ser encontrada numa publicação específica (relatórios do INIC da linha de Acção II do CIUL).

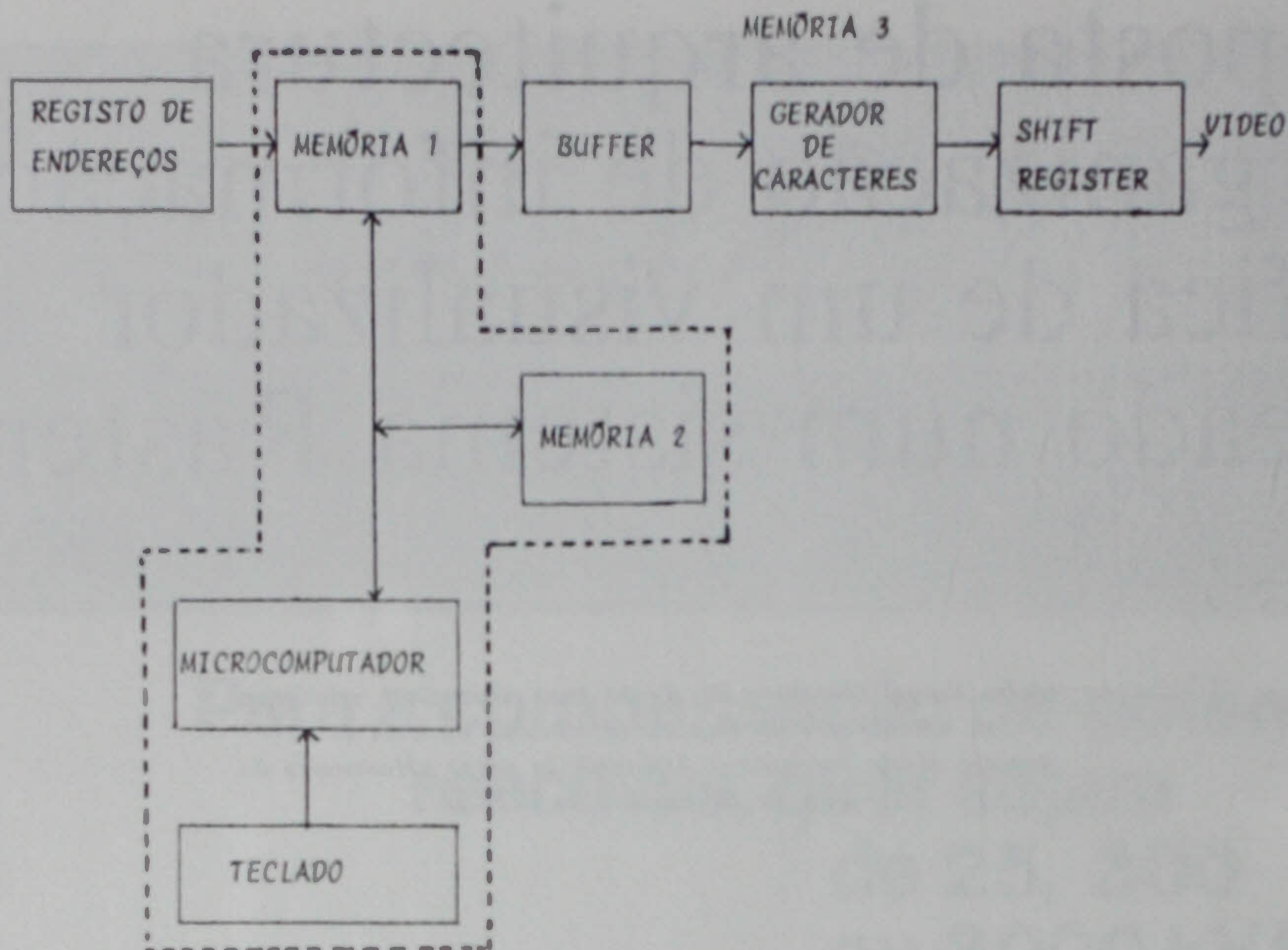


Fig. 3

1 — A memória *M1* será multiplexada de modo a poder pertencer ao espaço de endereçamento do microcomputador. *M2* pertence apenas ao espaço do microcomputador.

2 — A memória *M3* contém no seu interior a descrição adequada de símbolos destinada a permitir a formação de caracteres no écran.

estabelecer coordenadas absolutas ($x \rightarrow 0$ a 480 e $y \rightarrow 0$ a 240) para estes.

ARQUITECTURA GRÁFICA

Baseia-se na aplicação de três grupos de ideias que não constituindo em si campos originais (memórias pro-

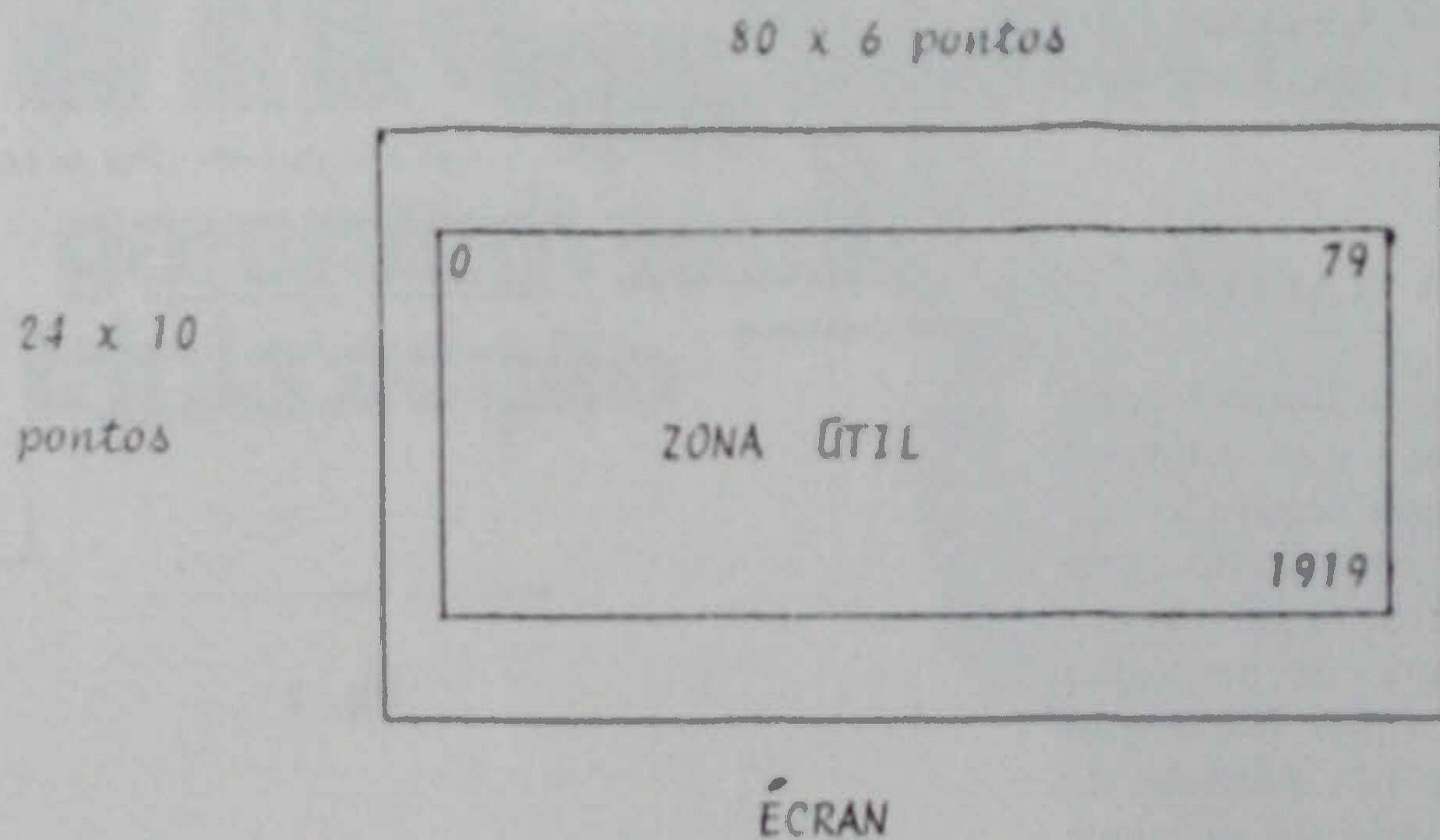
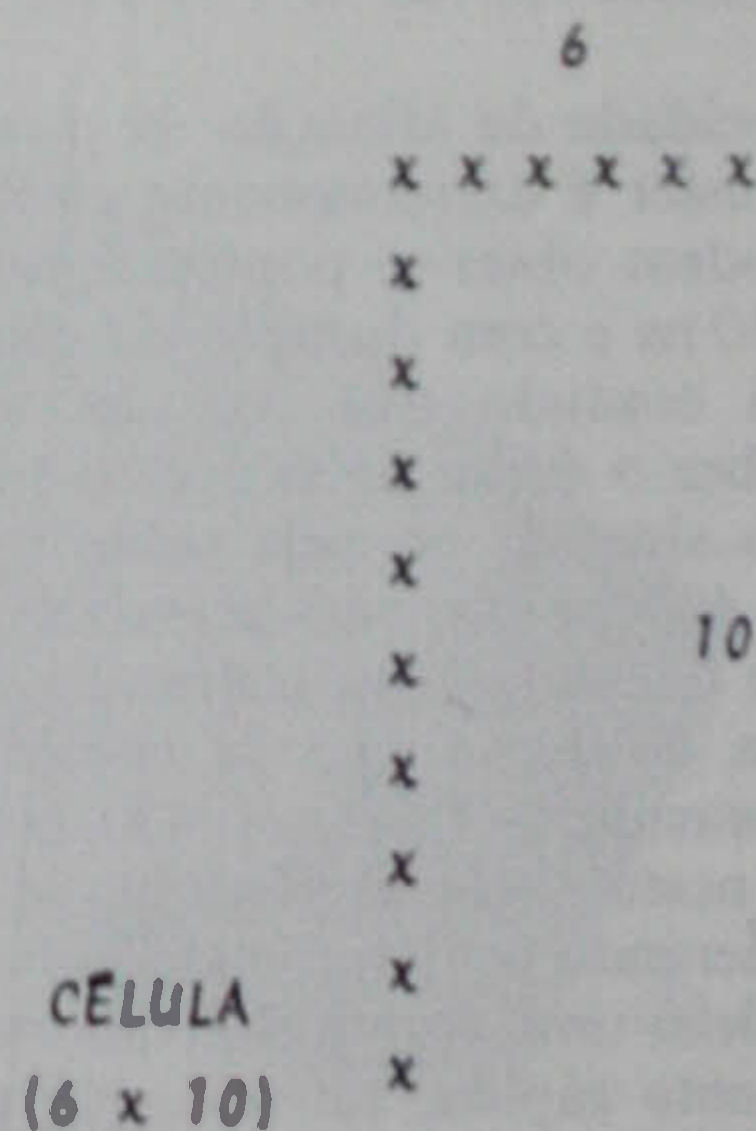


FIG 4



CÉLULA
(6 x 10)

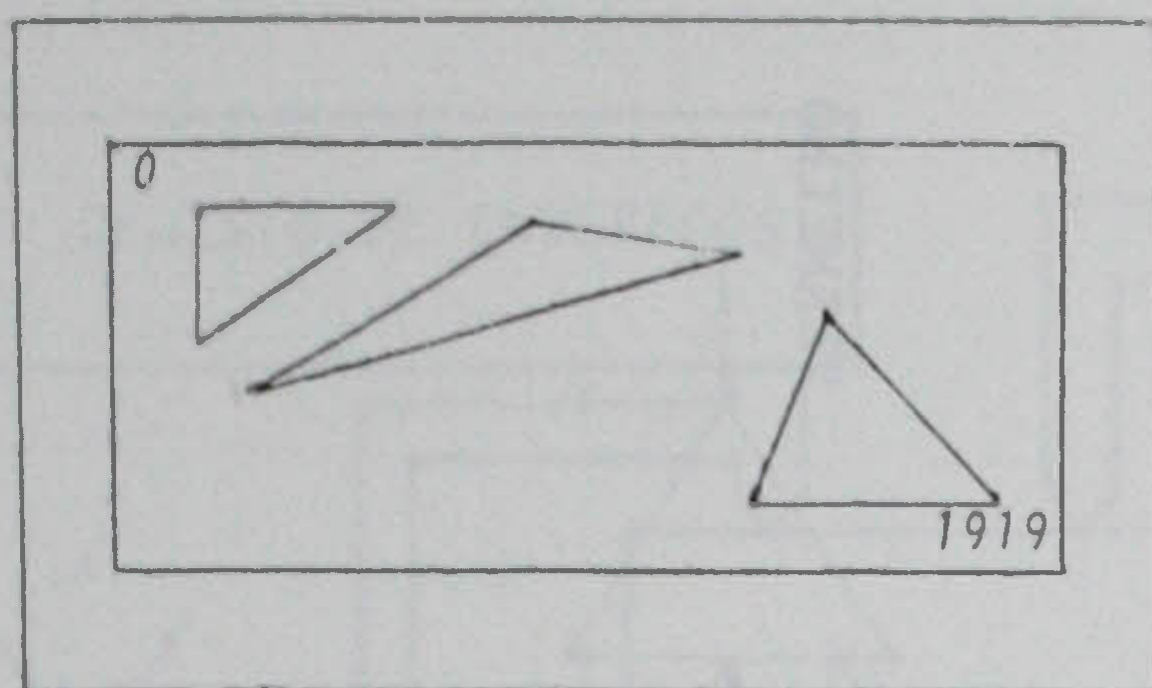
3 — Ao nível do microcomputador encontra-se residente o sistema de edição e comunicação do terminal (firmware).

Relembrar-se-á que o processo de formação de símbolos assenta na assimilação de 80×24 células formadas por 6×10 pontos a posições fixas no écran (fig. 4). Considerando os pontos individualmente poder-se-ão

gramáveis, integração de microprocessadores para gestão de informação, acção em tempo real), permitem conjuntamente uma solução que se afigura interessante.

Assim:

1 — Introduziu-se uma memória programável (RAM) multiplexada do mesmo modo que *M1* e *M2*



ECRAN E ZONA ÚTIL

Fig. 5

(fig. 3) no espaço anteriormente ocupado pelo gerador de caracteres (memória morta).

Esta memória é agora chamada através dos códigos existentes em *M1* ou acedida pelo microprocessador. Desde que tenha a dimensão suficiente pode permitir realizar sobre o ecrã qualquer conjunto de pontos (240×480).

Desde que se não considere a hipótese de aparecerem células com configurações repetidas, esta solução não permite poupar memória comparativamente com as configurações de bit map (fig. 5) uma vez que existirão 1920 células de 6×10 pontos que serão denominados Formas e que poderão ser todas diferentes definindo uma memória (Memória de Formas) com uma capacidade de $6 \times 10 \times 1920 \text{ bits} = 120 \text{ K bits}$. De facto, torna-se conveniente utilizar uma palavra de mais de 6 bits a fim de conseguir definir possibilidades de acções especiais (tempo real). Uma primeira solução foi ensaiada com 8 bits mas poderá ser conveniente uma utilização com uma palavra maior.

O modo de funcionamento desta parte do processador é similar à de um processador alfanumérico. Obviamente esta nova arquitectura poderá inscrever no ecrã qualquer tipo de símbolos incluindo os alfanuméricos.

2 — Como se referiu, o módulo do processador responsável pelo aparecimento da informação sobre o ecrã, (fig. 1 excluindo o microcomputador e a *M2*) deverá ser de execução extremamente rápida (80 ns cada ponto).

No que se refere ao tratamento e organização de ficheiros que definam a informação gráfica existente sobre o ecrã, assim como à capacidade de autonomamente se poder criar e interactivar com elementos gráficos (primitivas gráficas) utilizou-se intensivamente um microcomputador. Convém notar que a definição local de primitivas gráficas implica a implementação da interface que assegura a produção e carregamento em *M1* e *M3* dos «códigos máquina» do processador mais rápido, de modo a haver correspondência entre a maneira como estas são executadas e materializadas sobre o ecrã.

A multiplexagem de memórias (*M1* e *M3*) facilita grandemente os processos de transferência.

Também de maneira natural a comunicação com o exterior será apoiada sobre o microprocessador (interface série e paralelo).

3 — É conveniente permitir a manipulação de quantidades sucessivamente alargadas de informação gráfica em tempo real (ocultar, intensificar, mudar de cor, pulsar, substituir, etc.).

Mediante a descodificação de bits especiais existentes na memória *M1* e *M3* (fig. 6) ou identificando certos códigos existentes em *M1* ou comparando o conteúdo do registo de endereços com valores introduzi-

dos no microcomputador, é possível obter os resultados indicados.

Durante a descrição detalhada do funcionamento do processador gráfico introduzir-se-á o conceito de sub-programa gráfico de especial importância na acção de substituição.

DIAGRAMAS DETALHADOS DO PROCESSADOR GRÁFICO (Fig. 7 e Fig. 8)

O diagrama apresentado na fig 7 permite analisar microscopicamente o funcionamento do processador apresentado agora com uma extensão que permite a aplicação a monitores a cores. O gerador de caracteres é ainda incluído no sistema embora tal não seja necessário (nem conveniente) uma vez que as três memórias de Formas permitem criar qualquer tipo de grafismos.

Através dos multiplexores 1 e 2 o microcomputador acede às memórias do sistema *M1*, *M3*, *M3'*, *M3''*, *M3'''*. A multiplexagem faz-se automaticamente quando este selecciona uma zona de endereços (*Fxxx* e *9xxx*).

O registo de endereçamento permite, na ausência do microprocessador, a leitura sequencial dos códigos contidos na memória de Varrimento (*M1*) que vão servir como endereço inferior para acesso às memórias de Formas. Quatro bits adicionais provenientes de um contador, permitirão a extracção das dez linhas de cada Forma que irão permitir a definição do gráfico (fig. 9).

Este modo de endereçamento implicará que cada Forma não se encontre definida em endereços sequenciais da memória mas espaçadas do seguinte modo (fig. 10).

O endereço assim definido permite retirar da memória de Formas (*M3'*, *M3''*, *M3'''*) os sucessivos conjuntos de 8 pontos dos quais 6 serão carregados em *shift registers* para posterior saída série. Os restantes dois poderão ser usados para descodificação de acções (fig. 11) com efeito sobre os bits de cada linha de uma Forma.

Como se referiu uma maior dimensão da palavra das memórias de Formas permitirá diversificar o tipo de acções com maior liberdade.

A parte menos significativa do endereço da memória de Formas (código contido em *M1*) poderá também ser objecto de análise em tempo real permitindo desencadear acções que atinjam agora toda a Forma. A escolha dos códigos que serão activos poderá ser feita através do microcomputador carregando um registo para comparação (fig. 12).

Finalmente uma comparação similar com o conteúdo do registo de endereçamento permitirá iniciar (findar) uma acção a partir de uma qualquer posição de memória *M1* (fig. 13).

O diagrama apresentado na fig. 8 explicita mais em detalhe o modo como a arquitectura se articula.

A escolha da dimensão das memórias de Formas correspondem a uma opção de utilização de 50 % das formas possíveis no ecrã (1920) sem que se considerem repetições destas.

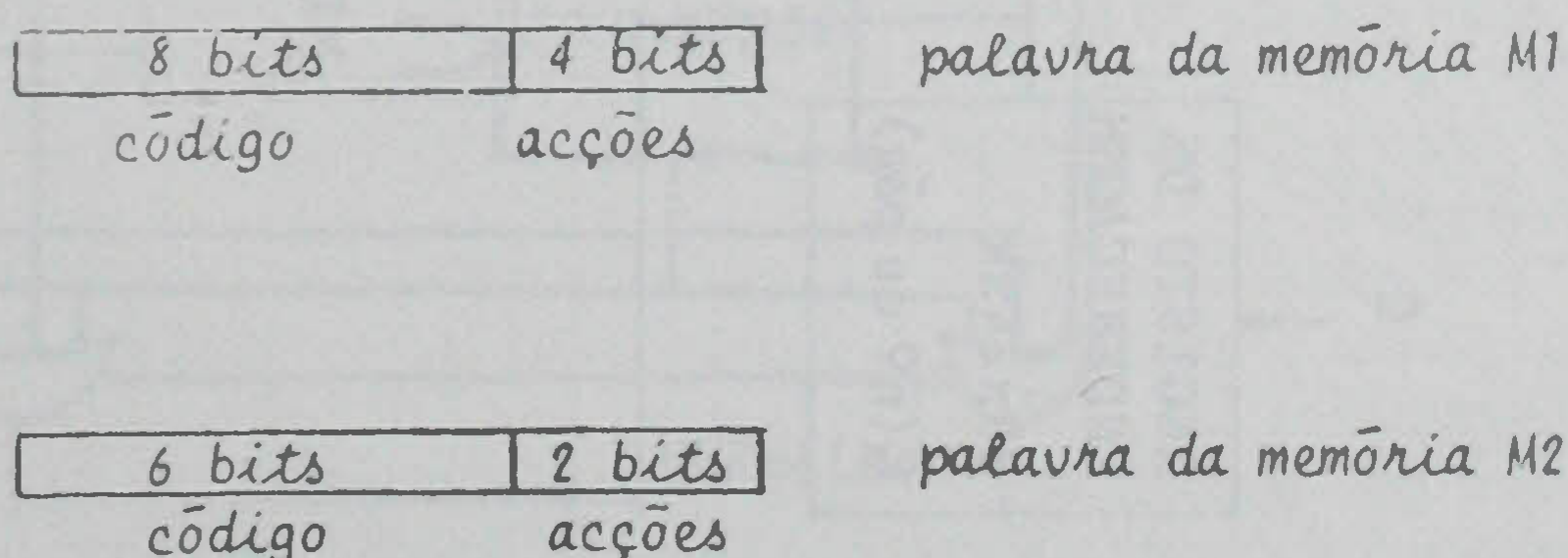


Fig. 6

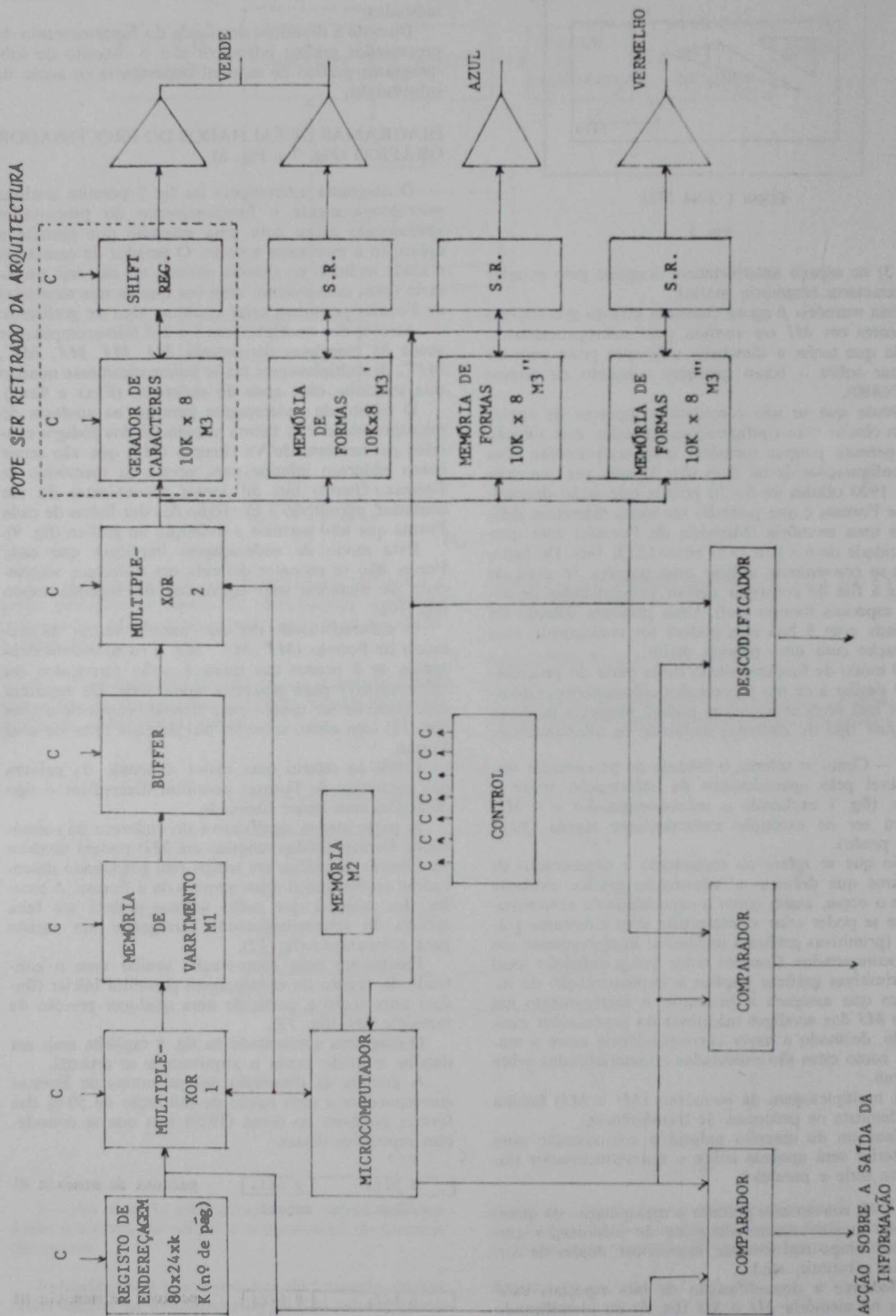


Fig. 7

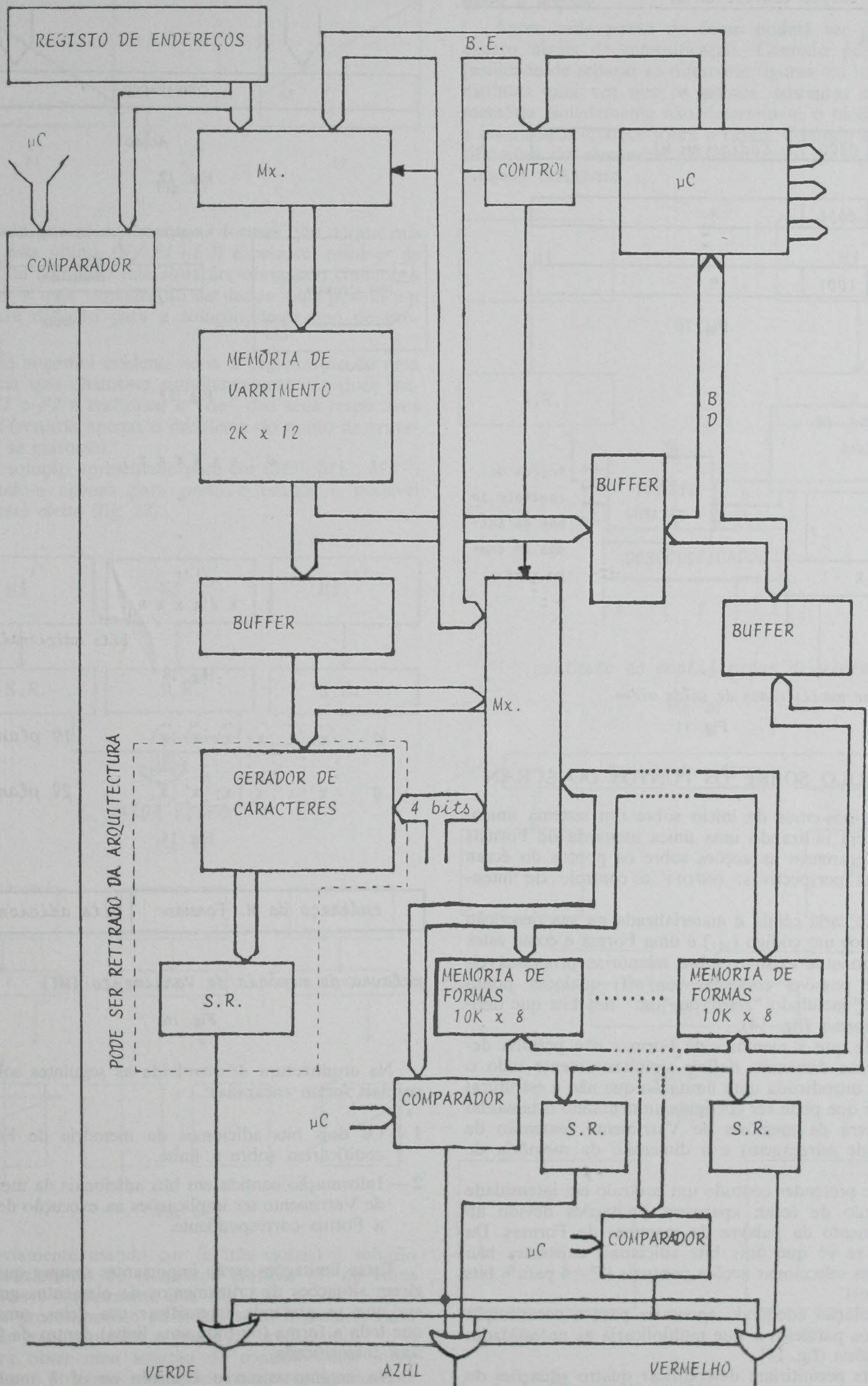


Fig. 8

DCBA	CÓDIGO CONTIDO EM M1
------	----------------------

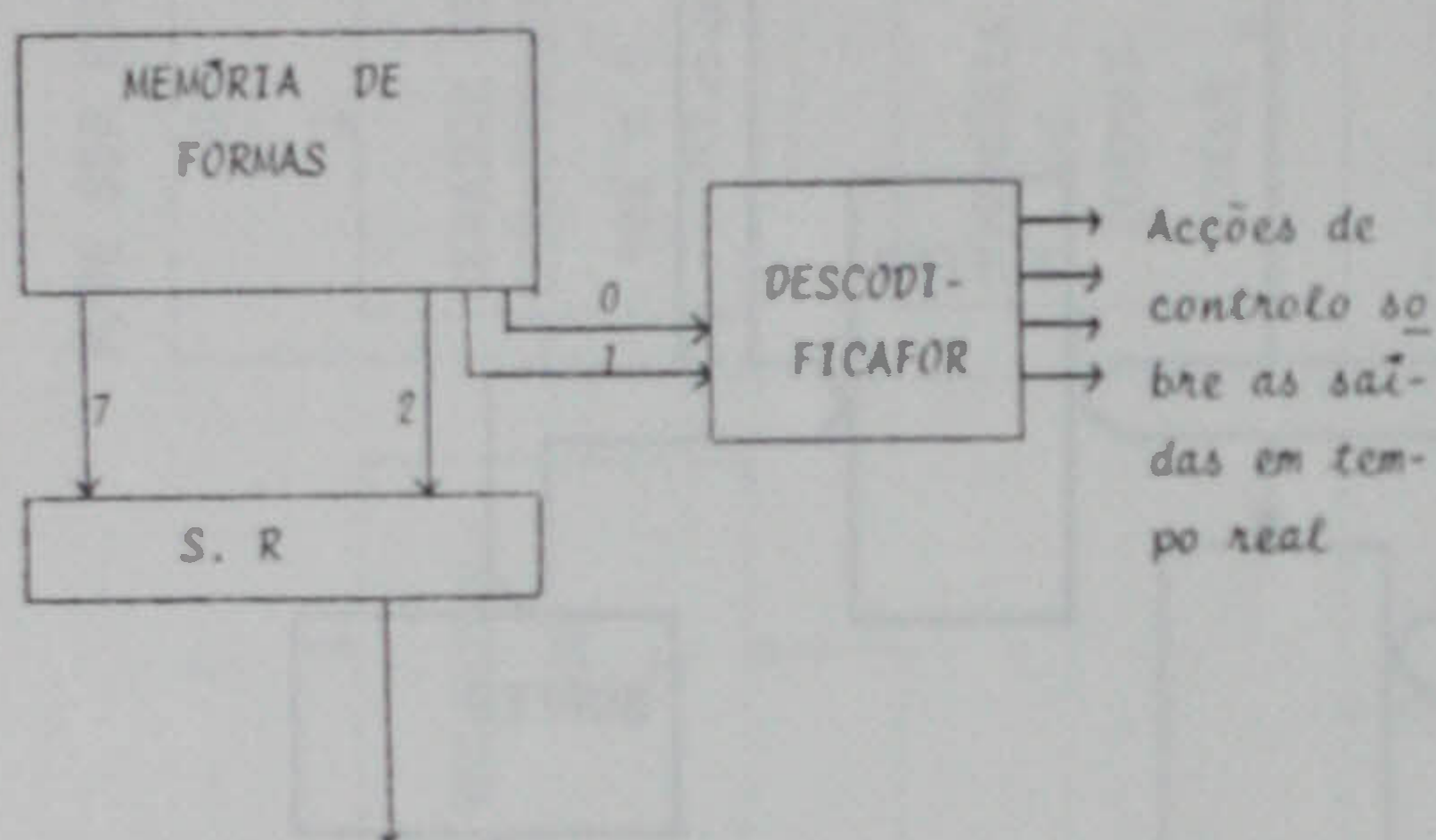
contador
de 10

Fig. 9

0000	Código em M1
0001	"
1001	"

Fig. 10

M3'



Ligação ao amplificador de saída vídeo

Fig. 11

CONTROLO SOBRE OS PONTOS DO ÉCRAN

Fixar-nos-emos de início sobre um sistema unicolor ou seja utilizando uma única memória de Formas e consideraremos as acções sobre os pontos do écran sob duas perspectivas: ON/OFF e controlo de intensidade.

Como cada célula é materializada na sua descrição gráfica por um código (M_1) e uma Forma e como estes dois elementos existem sobre memórias programáveis (RAM) é possível controlar (on/off) qualquer ponto do écran, incluindo "zero" ou "um" nos bits que definem a forma (fig. 14).

Desde que a memória de Formas não permita devido à sua dimensão definir simultaneamente todo o écran, é introduzida uma limitação que não é estrutural uma vez que pode ser corrigida aumentando o tamanho da palavra da memória de Varrimento (extensão de espaço de endereçamento) e a dimensão da memória de Formas.

Se se pretender contudo um controlo em intensidade por ponto de écran aparecem limitações devido ao cumprimento da palavra da memória de Formas. Da fig. 14 se vê que dois bits adicionais existentes não permitem seleccionar acções pontuais ($2^2=4$ para 6 bits existentes).

A solução adequada apontaria para a constituição de planos paralelos o que multiplicaria as necessidades de memória (fig. 15).

2 bits permitiriam decodificar quatro situações do ponto (10, nada, normal, brilhante, muito brilhante por exemplo).

endereço vindo da memória M1

microcomputador

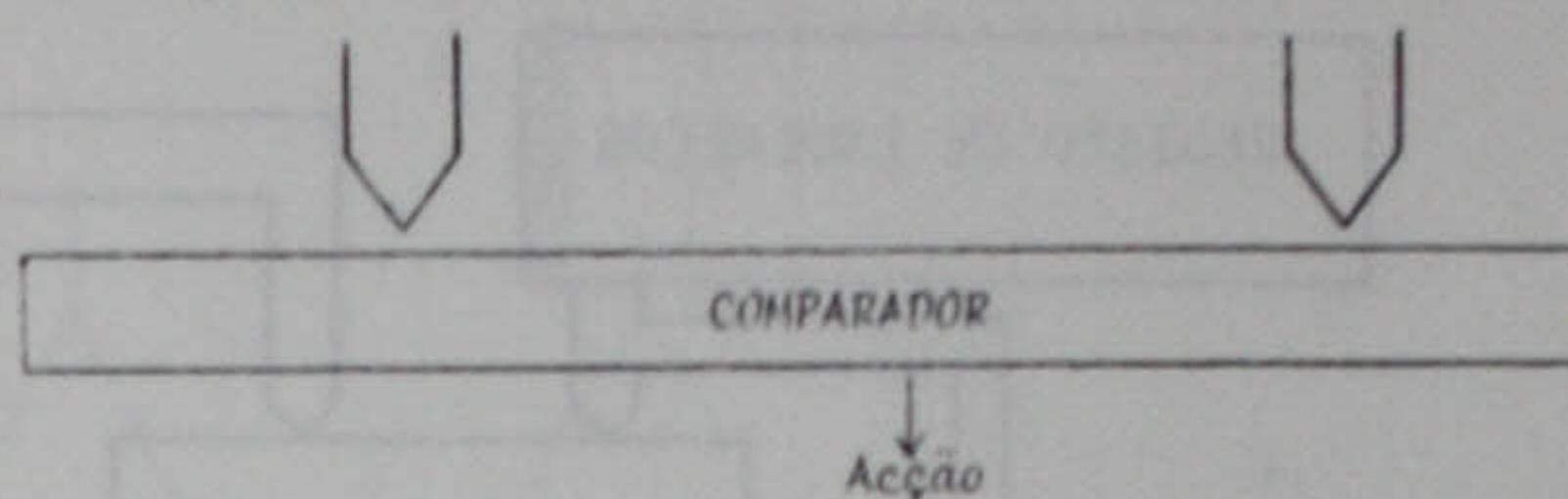


Fig. 12

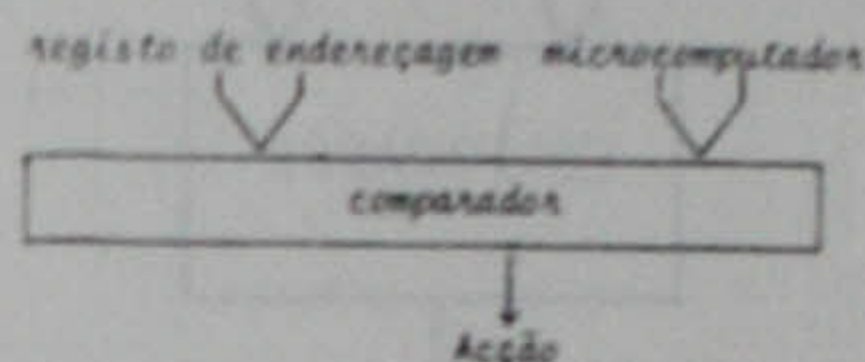
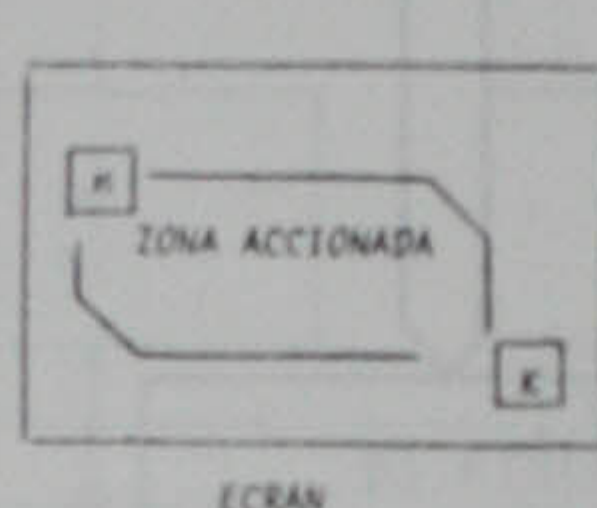


Fig. 13

0 x x x x x x . .
.
.
.
9 x x x x x x . .
bits adicionais

Fig. 14

0 x x x x x x x 1º plano
0 x x x x x x x 2º plano

Fig. 15

endereço da M. Forma	bits adicionais
----------------------	-----------------

palavra de memória de Varrimento (M_1)

Fig. 16

Na arquitectura desenvolvida as seguintes soluções parciais foram encaradas:

- 1 — Os dois bits adicionais da memória de Formas codificarem sobre a linha.
- 2 — Informação contida em bits adicionais da memória de Varrimento ter implicações na execução de toda a Forma correspondente.

Estas limitações serão importantes sempre que surgirem situações de cruzamentos de elementos gráficos em que se pretenda intensificar um deles, uma vez que toda a forma (ou toda uma linha) dentro da forma será intensificada.

No entanto como o controlo on/of é total não haverá dificuldade estrutural de resolução do problema dos cruzamentos (fig. 17).

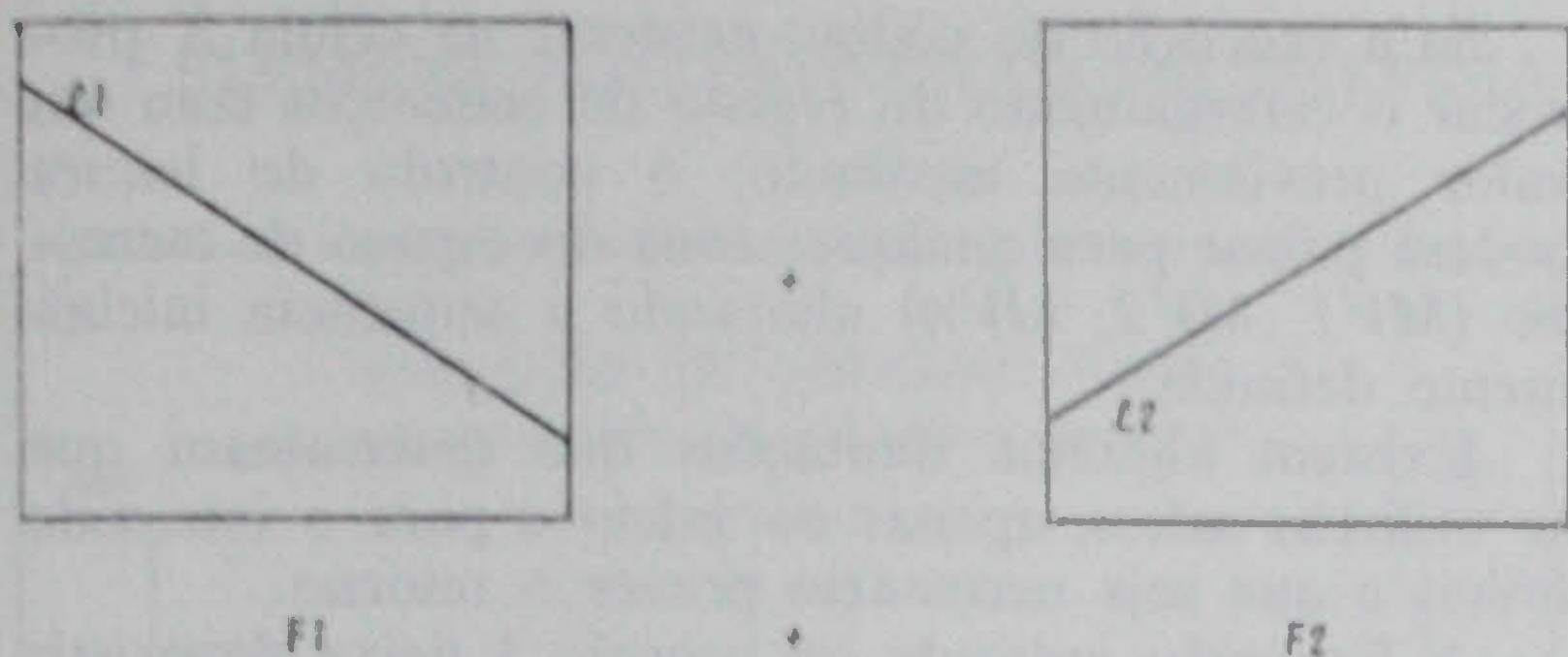


Fig. 17

Desde que se conheçam as formas que deram origem a esta última ($F=F1+F2$) é possível resolver de um modo completo esta situação, obrigando contudo a recorrer a uma organização de dados mais pesada e a firmware dirigido para a solução deste tipo de problemas.

Uma sugestão evidente seria a organização de uma memória que chamasse simultaneamente as duas formas $F1$ e $F2$ e realizasse o "ou" dos seus respectivos pontos (restaria apenas o problema do ponto de cruzamento se existisse).

Na solução apresentada para cor ($M3'$, $M3''$, $M3'''$) utilizando-a apenas para preto e branco é possível obter este efeito (fig. 18).

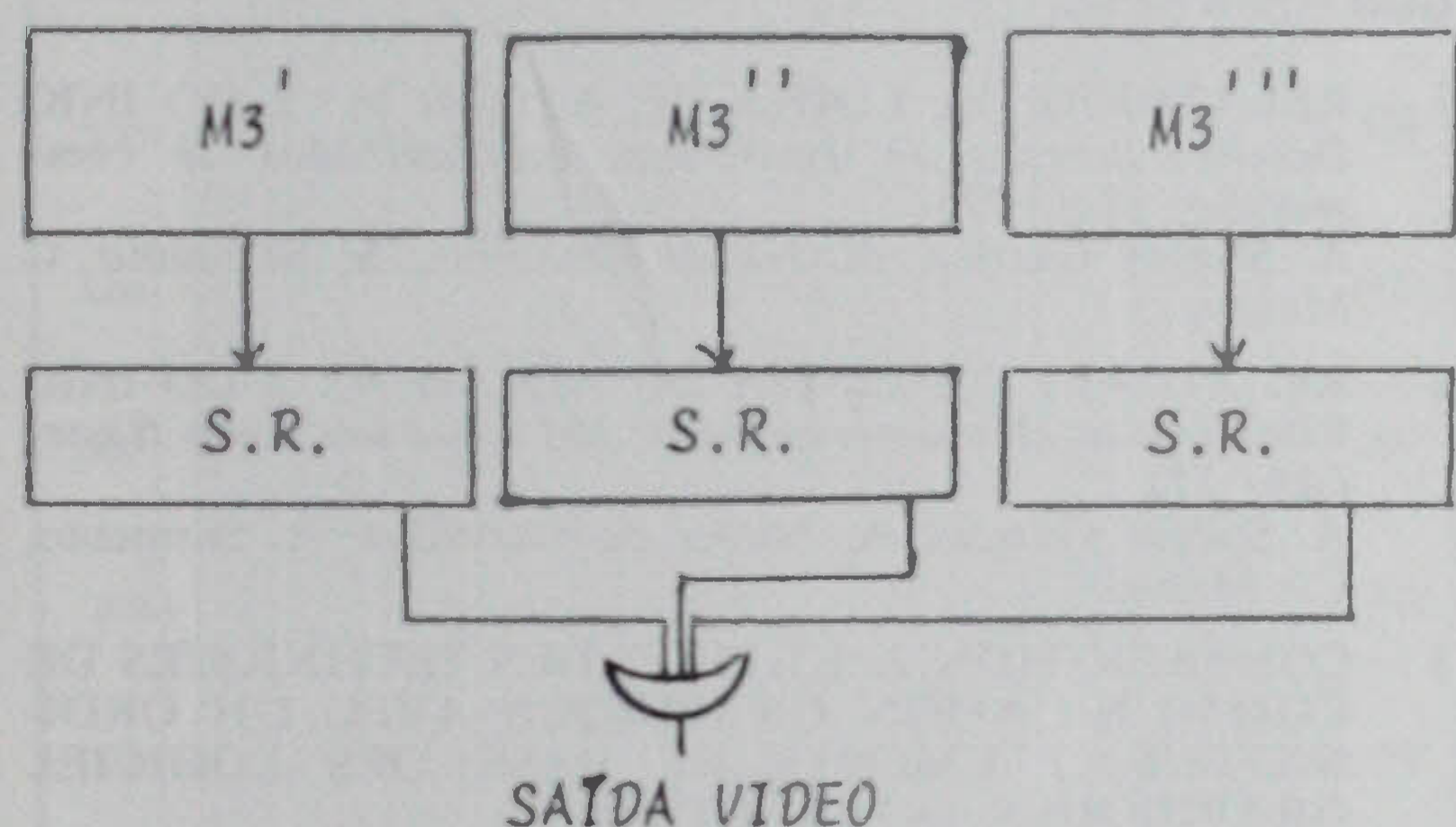


Fig. 18

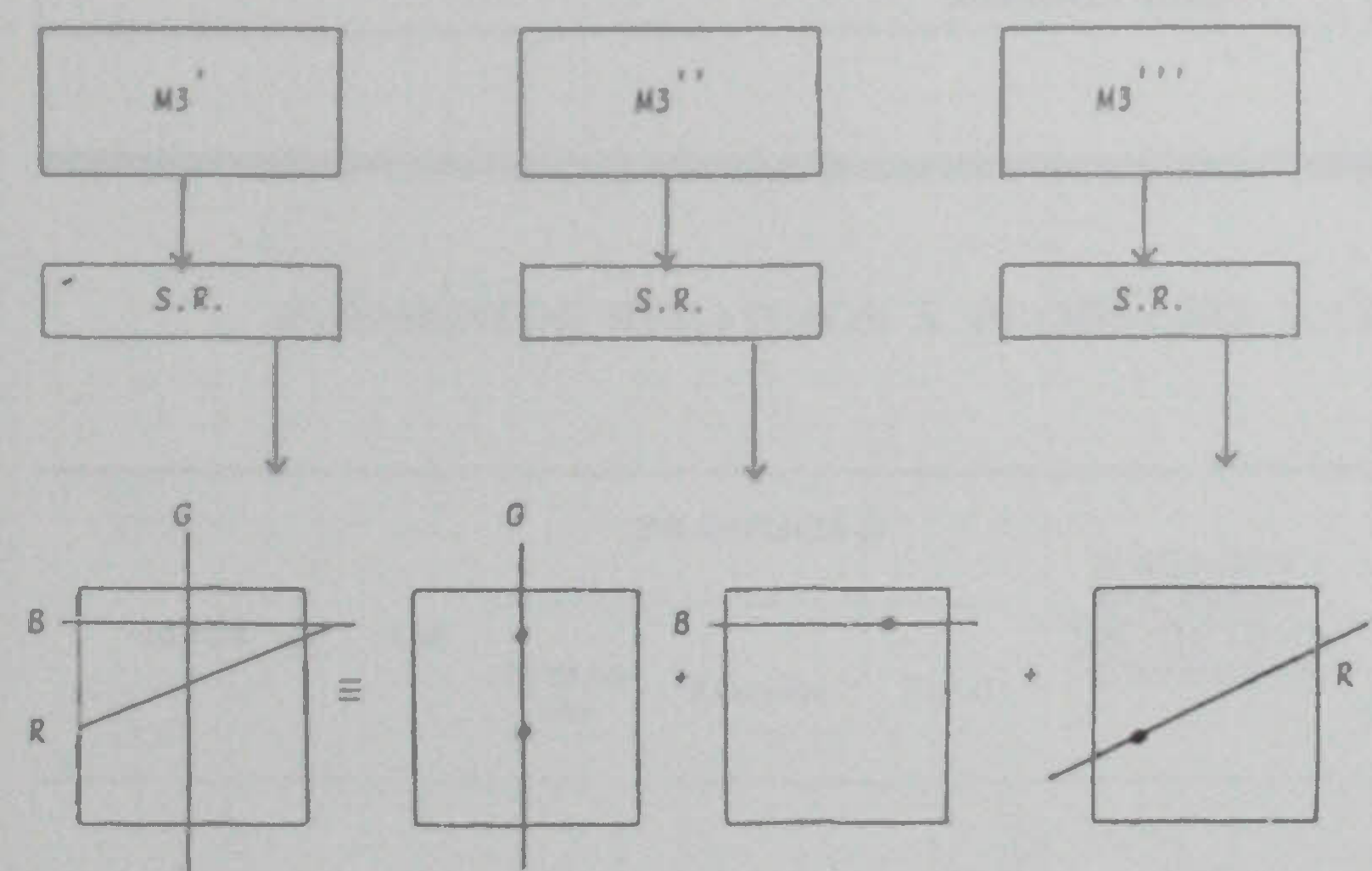


Fig. 19

Obviamente usando cor (e três canais) a solução para cruzamento de elementos gráficos (3) de cores diferentes é exactamente esta (fig. 19).

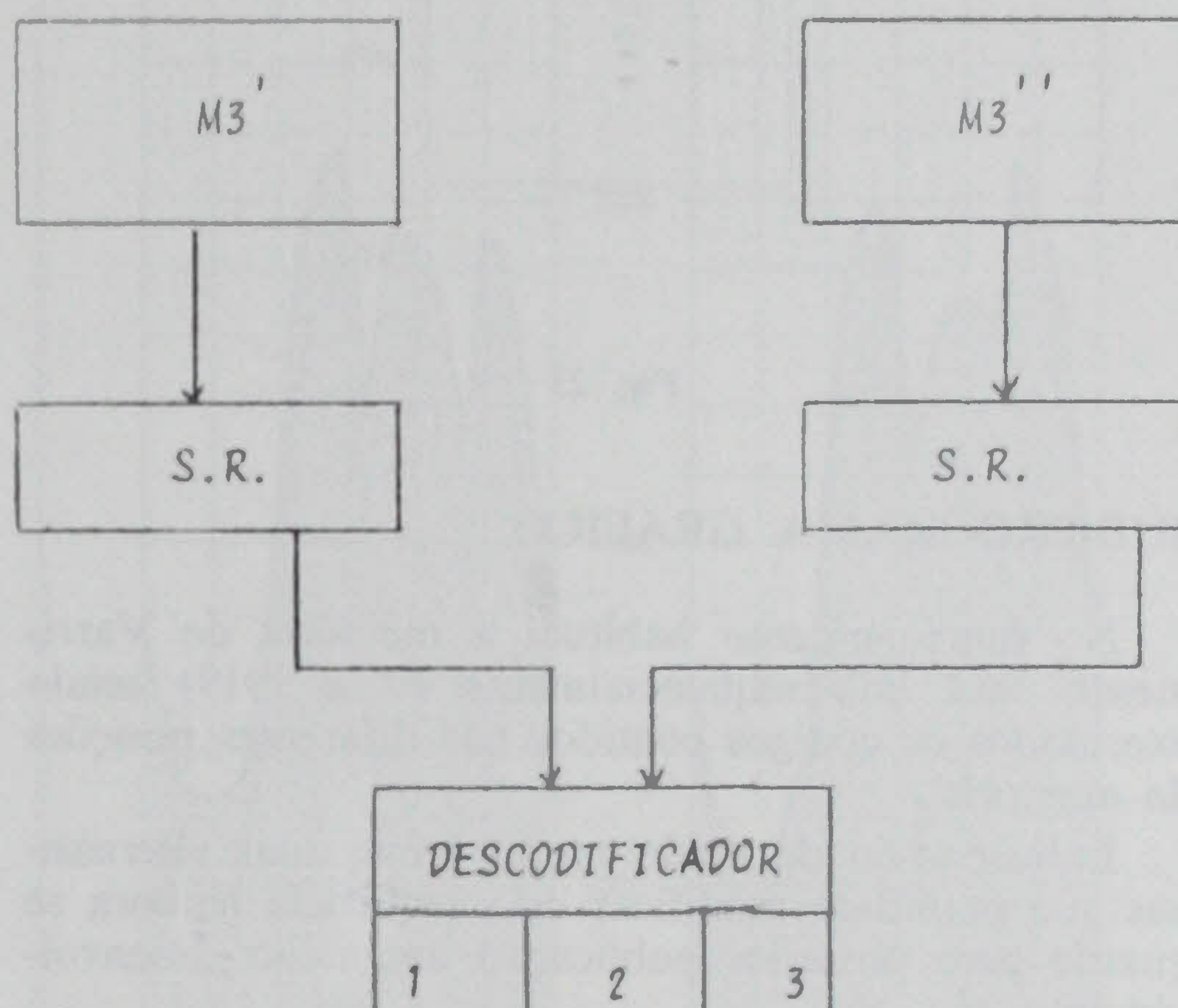
Um processamento adicional seria necessário para permitir tratar também o ponto de cruzamento.

Para obter uma solução de controlo total sobre cada ponto do écran será suficiente alterar ligeiramente a configuração expressa na fig. 18.

Exemplifica-se apenas com duas Memórias. Utili-

zando três memórias ($M3'$, $M3''$, $M3'''$) poder-se-iam obter 8 acções.

Agora cada ponto do écran poderá ser sujeito a quatro níveis de intensificação. Contudo perde-se a facilidade de separar as diferentes figuras em memórias distintas uma vez que os pontos existentes em cada memória isoladamente não determinam o modo como a informação aparece sobre o écran. Contudo as intensificações de elementos gráficos serão agora completamente definíveis.



controlo do amplificador de video

Fig. 20

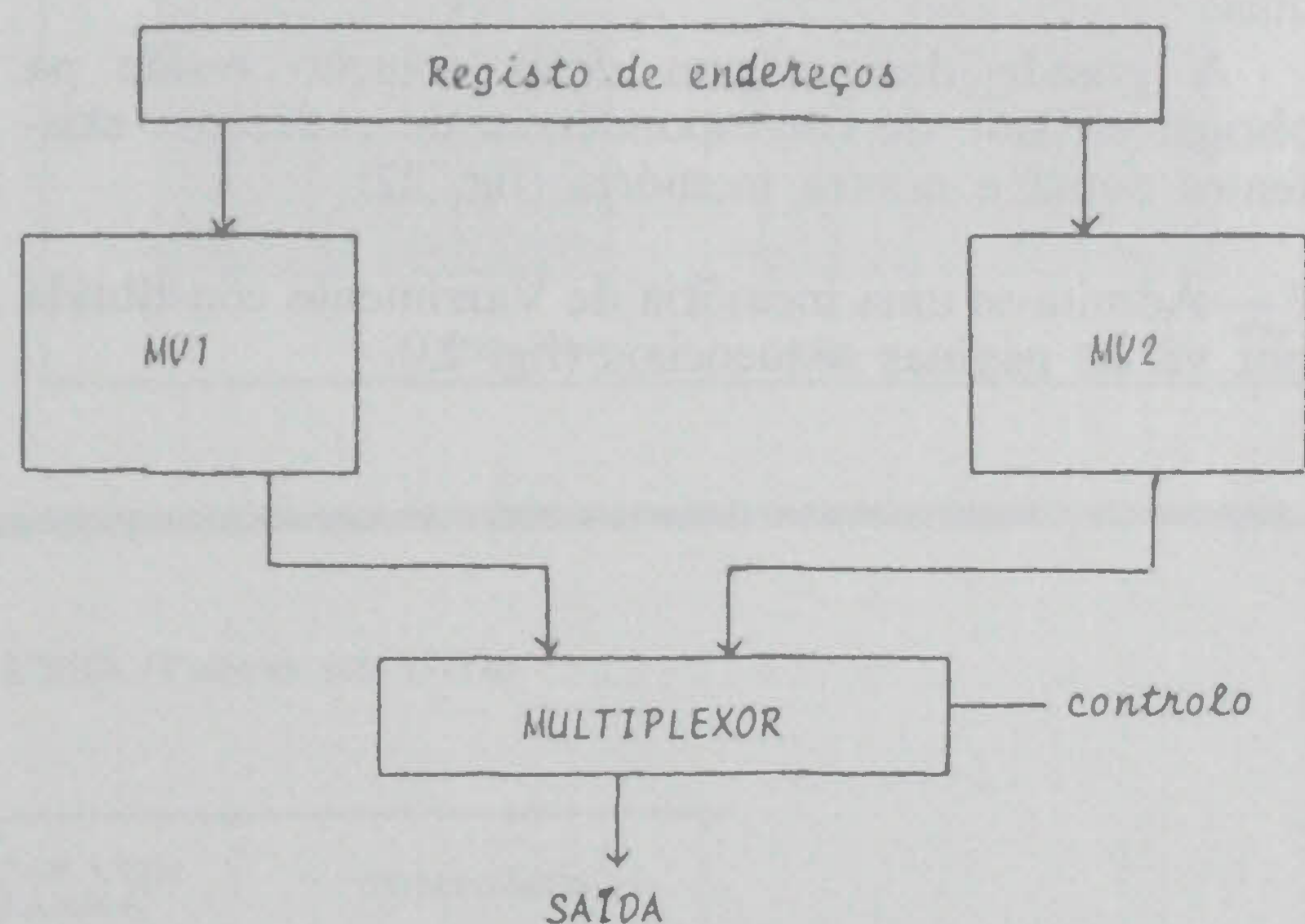
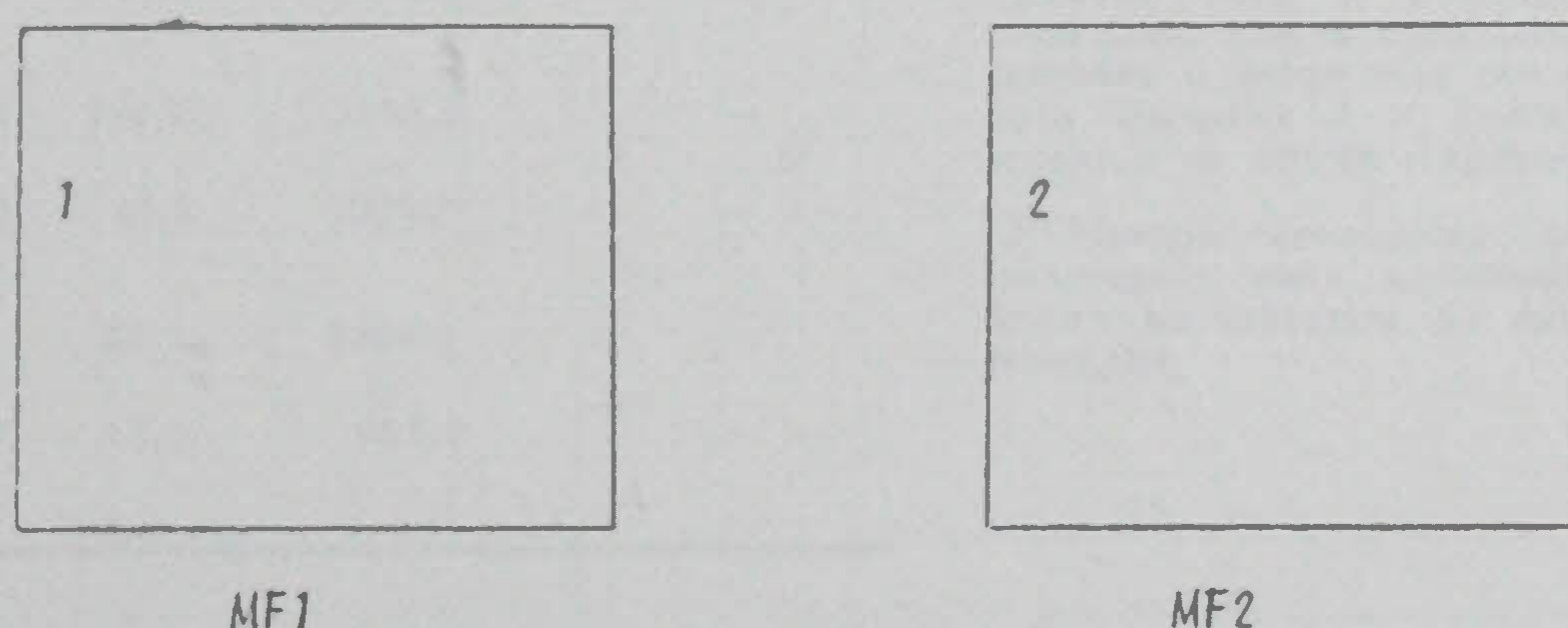


Fig. 21



1 e 2 têm o mesmo endereço

Fig. 22



Fig. 23

SUB-PROGRAMA GRÁFICO

No funcionamento habitual a memória de Varriemento será lida sequencialmente (0 a 1919) sendo executados os códigos contidos nas diferentes posições de memória.

Indicar-se-ão de modo muito breve duas alternativas que permitem modificar esta sequência embora se guarde para posterior publicação um maior desenvolvimento.

1 — Admitindo duas memórias de Varrimento funcionando em paralelo, apenas com as saídas multiplexadas (fig. 21) é possível transferir as saídas via multiplexor executando assim segmentos existentes em cada uma.

A grande desvantagem desta solução reside na obrigatoriedade de correspondência de endereços existentes numa e noutra memória (fig. 22).

2 — Admita-se uma memória de Varrimento constituída por várias páginas sequenciais (fig. 23).

Se a execução do código existente na célula X provocar o carregamento do registo de endereços com um valor previamente escolhido, o controlo de leitura poderá passar para qualquer zona do espaço de memória ($MV1$, $MV2$, MVn) alterando a sequência inicialmente definida.

Existem algumas limitações que determinam que se utilizem saltos, apenas no início e para o início de linhas e que seja necessário prever o retorno.

A limitação indicada na solução 1 deixa de existir nesta arquitectura.

Uma última nota referente à organização de informação gráfica. O seu volume e especificidade levaram-nos também a transferir para outra publicação, complementar da actual, a sua descrição.

Um agradecimento final a toda a equipa de Hardware do Departamento de Informática, em especial aos seus técnicos, assim como à Secretária do Departamento, Conceição Gaiolas, pelo cuidado e brevidade com que permitiu a saída dactilografada do manuscrito.

Manifestamos também o nosso especial apreço a C. M. da Silva e A. Cerveira pelas sugestões referentes à introdução na arquitectura de memórias programáveis (Formas), controlo em tempo real e discussão detalhada do projecto.

BIBLIOGRAFIA

- 1 — RELATÓRIO DA LINHA DE ACÇÃO N.º 2 DO INIC *Desenvolvimento de periféricos especializados de computador (Set. 77)*
A. Steiger Garção, A. Neves Carvalho, A. Sarmento, C. Manso.
- 2 — RELATÓRIO DA LINHA DE ACÇÃO N.º 2 DO INIC *Visualizador alfanumérico realizado sobre um écran Raster (Ab. 77)*
A. Steiger Garção, A. Neves de Carvalho, A. Sarmento, C. Manso.
- 3 — CONTRIBUTION A L'ETUDE DES TECHNIQUES DE COMMUNICATION GRAPHIQUE AVEC UN ORDINATEUR. ELEMENTS DE BASE DES LOGICIELS GRAPHIQUES INTERACTIFS.
Michel Lucas
- 4 — AN INTRODUCTION TO MICROCOMPUTERS
Adam Osborne