

PCS 3216

Sistemas de Programação

João José Neto

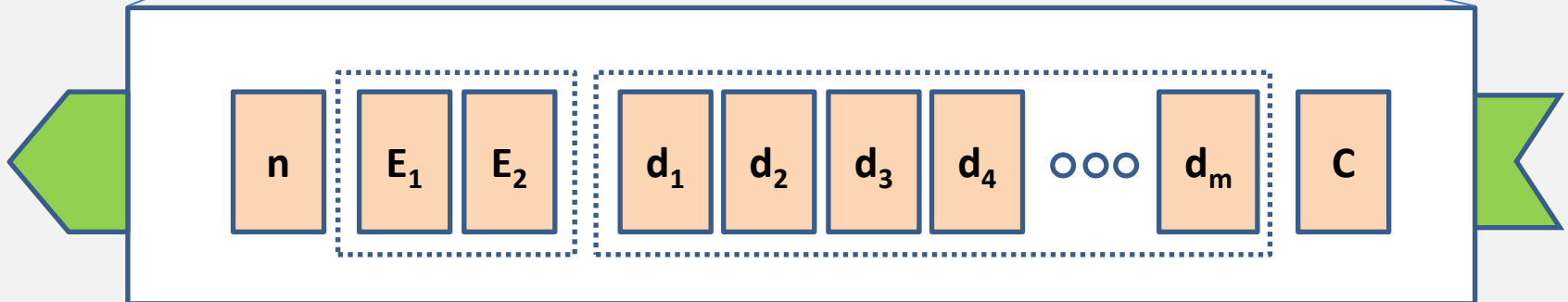
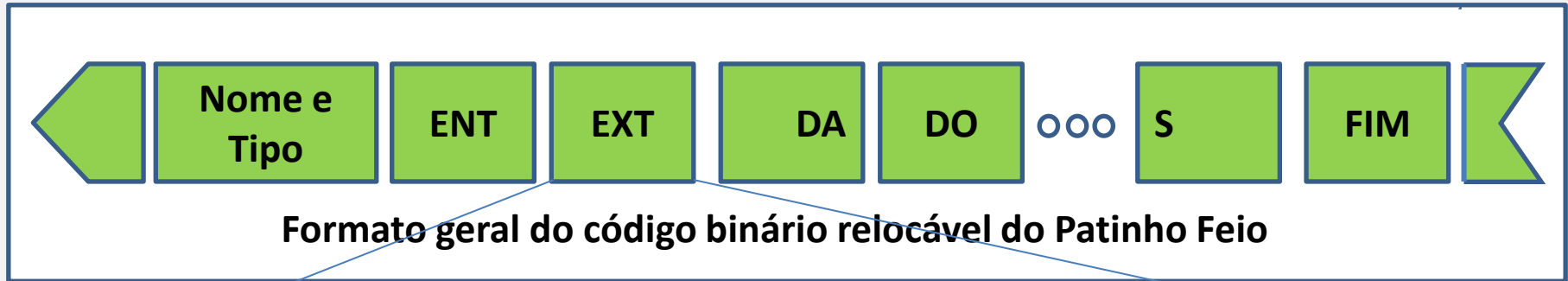
Aula 21 - Projeto Completo de um Ligador-Relocador

Apresentação

- Este material é uma versão ligeiramente modificada da documentação histórica manuscrita do projeto de um ligador-relocador, desenvolvido para o computador Patinho Feio por volta de 1972.
- Destina-se a exemplificar em detalhe e servir de modelo para a criação de um ligador-relocador, módulo essencial de um sistema de programação para computadores de pequeno porte.
- Deseja-se que o aluno faça as devidas adaptações para que o software que vier a ser produzido sirva para uso pelo processador virtual MVN, no sistema de programação que está sendo desenvolvido na disciplina PCS 3216.

FORMATO DO CÓDIGO RELOCÁVEL

FORMATO GERAL



$n = m - 1$ (Obs.: isto sinaliza que é inadequado o uso de blocos vazios)

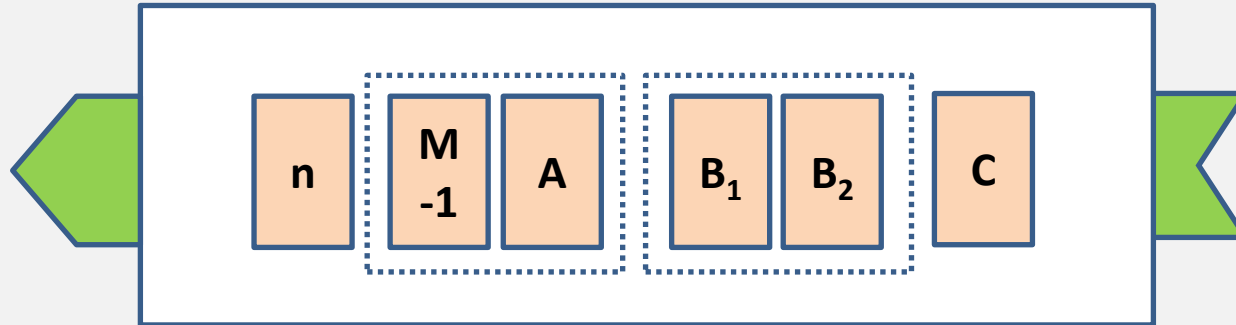
m = número de dados da sequência de dados $d_1 \dots d_m$

C = checksum (complemento de 2 da somatória (módulo 256) dos bytes anteriores)

(Obs.: a soma de todos os bytes do bloco, incluindo n e o checksum, deve ser nula)

Ver, ao final da sequência de figuras, os significados dos símbolos nelas utilizados.

Bloco de NOME e TIPO



Formato geral do blocos de *Nome e Tipo*

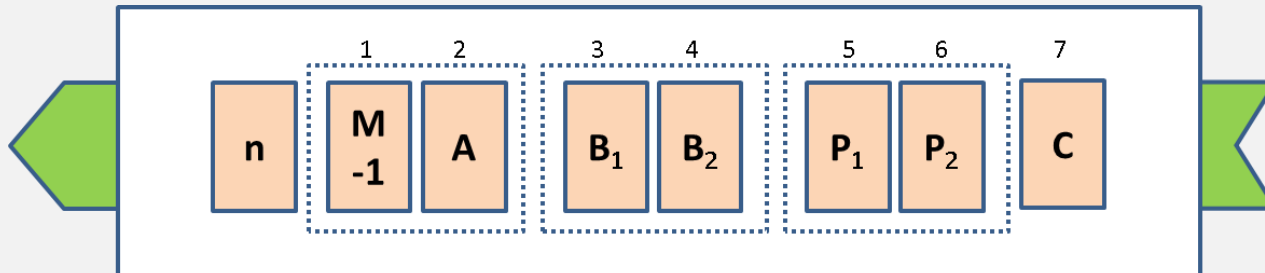
n = número de bytes seguintes no bloco (no caso acima, $n=5$)

NOME E TIPO \Rightarrow um ou nenhum bloco:

- se o código tiver um bloco de NOME, é relocável, \therefore OK.
- se não tiver, será absoluto, \therefore mensagem de erro.

É possível incluir no bloco “NOME” uma informação sobre o comprimento total do programa (número de bytes ocupado pelo programa ou subrotina).

Neste caso, “NOME” ficaria conforme a figura seguinte, onde (P1,P2) contém [em 2 bytes] o número de bytes ocupado pelo programa ou subrotina.

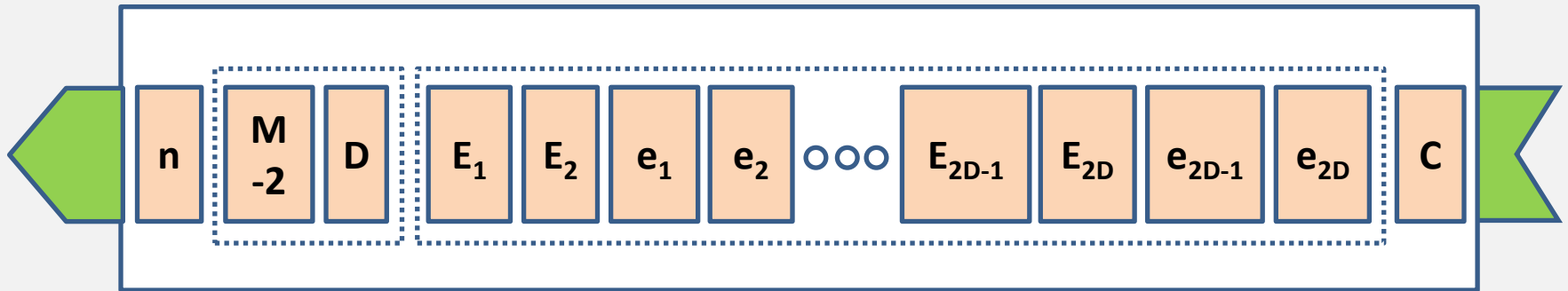


Formato do bloco de *Nome e Tipo*,

[incluindo o comprimento do programa] ($n = 7$; $m = 1$)

projeto do ligador-relocador para o patinho feio - J.J.Neto

Bloco ENT



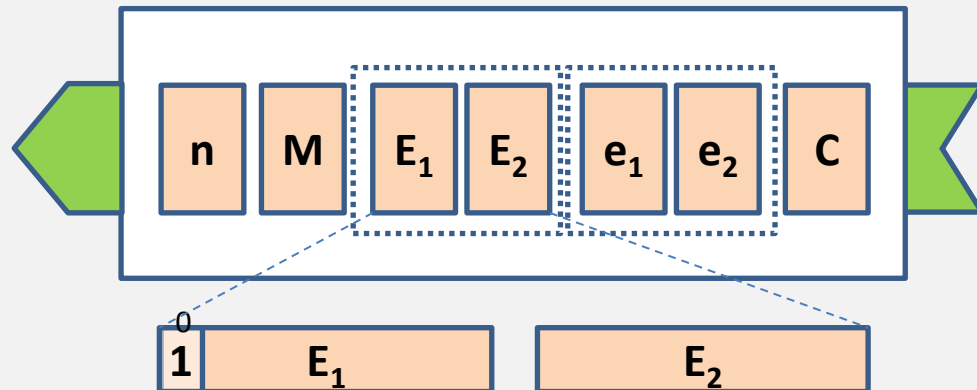
Formato geral do bloco ENT

ENT

=> duas hipóteses:

- Ou usar como está na figura (um só bloco, para todos os ENTRY (mais trabalhoso)).
- Ou fazer um bloco de ENT para cada declaração ENT no programa (mais fácil) .

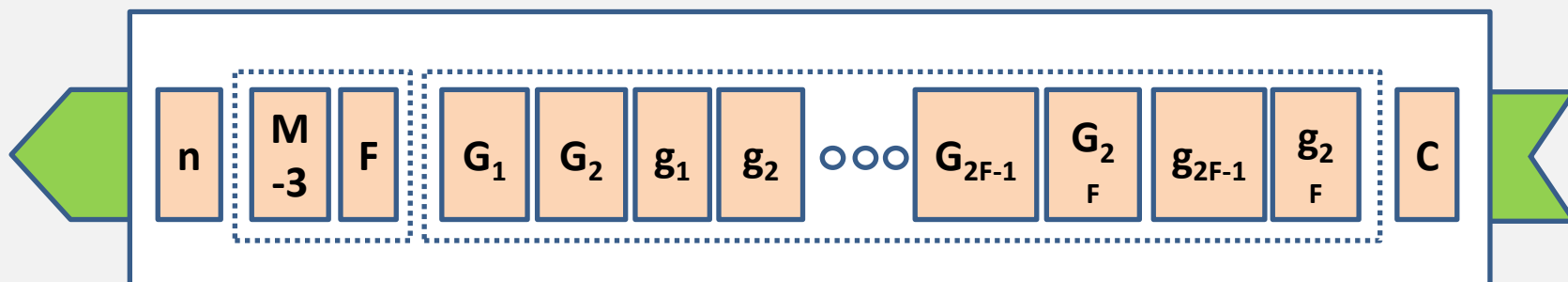
Neste caso, o ligador-relocador deve garantir que a sequência dos blocos esteja correta, e um bloco de ENT ficaria com o formato seguinte:



Formato alternativo do bloco ENT:

$n = 6; m = -2$

Bloco EXT



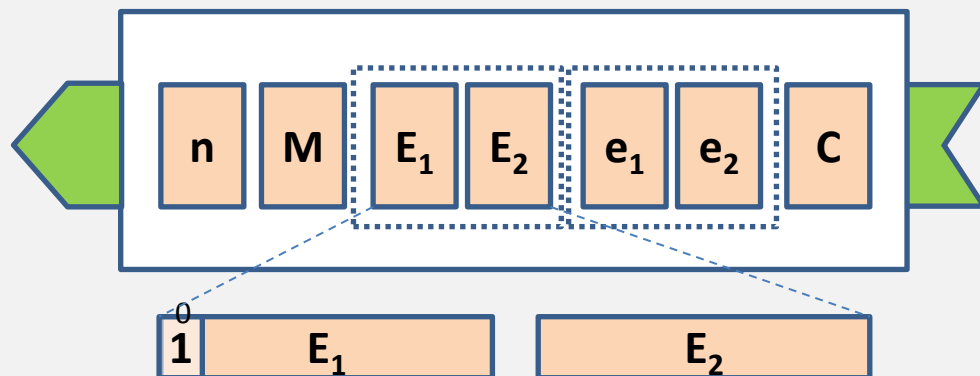
Formato geral do bloco EXT

EXT

=> duas hipóteses:

- Ou usar como está na figura (um só bloco, para todos os EXT (mais trabalhoso)).
- Ou fazer um bloco de EXT para cada declaração EXT no programa (mais fácil) .

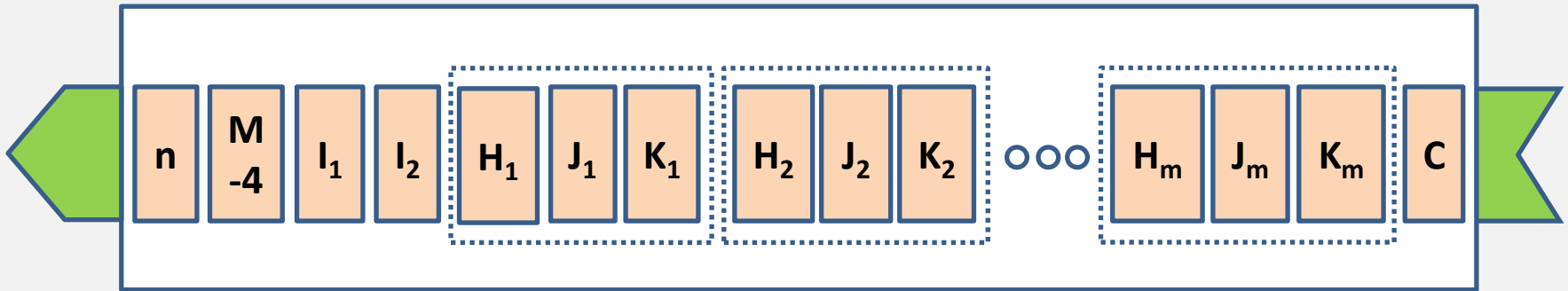
Neste caso, o ligador-relocador deve garantir que a sequência dos blocos esteja correta, e um bloco de EXT ficaria com o formato seguinte:



Formato alternativo do bloco EXT:

$n = 6$; $m = -3$

Bloco de DADOS

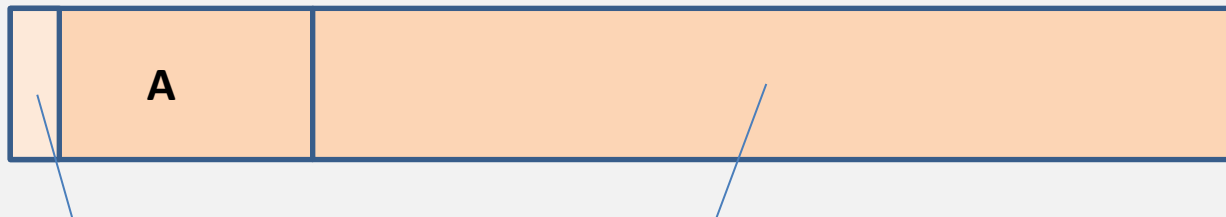


Formato geral dos blocos de DADOS

$$n = m-1$$

DADOS

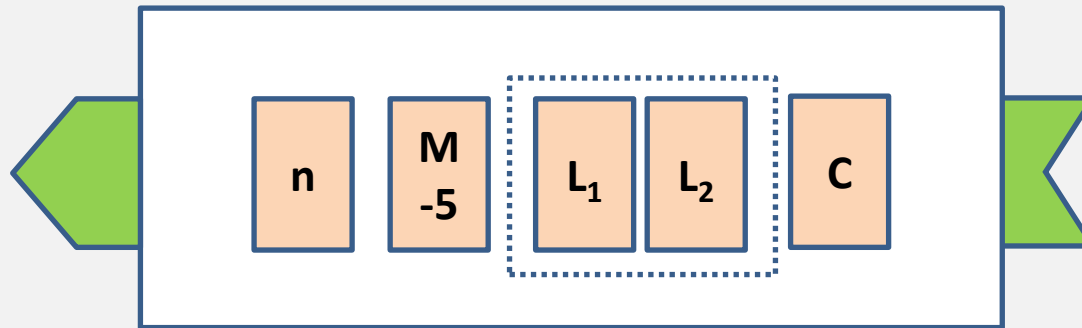
- Usar tantos blocos quantos forem necessários para comportar todo o código do programa.
- [Para simplificar o processamento deste tipo de blocos, e também para tornar menos difícil a tarefa de decifrar o código relocável] em um bloco devem comparecer [apenas] informações completas.
- Por isso, não convém separar uma instrução em dois blocos diferentes.



Def./Indef. Def. => endereço de A

Indef. => endereço da subrotina chamadora de A

Bloco FIM



Formato geral do bloco de FIM

FIM

-É único para cada rotina ou programa principal, separadamente compilados.

Códigos usados nos blocos

n	Número de palavras de 8 bits que vem a seguir, incluindo o checksum (-1 indica que é um bloco de Nome/Tipo)
A	0 => programa principal; $\neq 0$ => subprograma
B₁, B₂	Nome do programa/subprograma empacotado em dois bytes (usando a rotina PACK)
C	Complemento de 2 da soma (módulo 256) dos bytes do bloco
D	Número de entry points
E_i, E_{i+1}	Nome do (i+1/2)-ésimo entry point (i=1,3,5...2*D)
e_i, e_{i+1}	Endereço relocável do (i+1/2)-ésimo entry point
F	Número de externals
G_i, G_{i+1}	Número do (i+1/2)-ésimo external (i=1,3,5...2*F)
g_i, g_{i+1}	Nº. pelo qual ele vai ser referenciado no programa (é o end. relocável na tab. de subrotinas do ligador-relocador)
H_i	0 => abs., de 1 byte; 1 => relocável, de 2 bytes; -1 => relocável, de 2 bytes em rel. à tab. de subr. do ligador-relocador
I₁, I₂	Origem relocável do conjunto de bytes que vêm a seguir
J_i, K_i	i-ésimo byte do código: se $H_i = 0 \Rightarrow J_i =$ dado absoluto de 1 byte, $K_i = 0$ se $H_i = 1 \Rightarrow J_i =$ primeiro byte relocável, $K_i =$ segundo byte relocável
L₁, L₂	Endereço de execução do programa principal, ou então =0, se for subrotina.
M	Sequência dos blocos: -1 nome/tipo (um); -2 ENT (um) ; -3 EXT (um); -4 dados (o suficiente); -5 FIM (um); Entre um bloco e outro, são incluídos 4 bytes nulos para permitir a localização do início do bloco por parte do usuário.

COMO O LIGADOR-RELOCADOR DEVE FUNCIONAR

Como funciona o Ligador-relocador

1. Ler um bloco na fita. Testar na leitura o checksum.
Se não bater => Mensagem de erro, aguardar releitura do bloco .
Testar se o 1º bloco é do tipo “NOME”.
Se for => testar A para descobrir se é programa principal ou subrotina.
Se for subrotina => erro (o 1º código deve ser do programa principal).
Se for prog. principal => OK => Guardar nome comprimento numa tabela
Testar se (CR (Constante de relocação) + comprimento do programa)
invade a área protegida. Guardar a soma em uma variável auxiliar CR1.
Se invadir => mensagem de erro (overflow de memória) (irrecuperável)
Se não invadir => OK => continua o processo.
2. Ler outro bloco na fita. Deve ser um bloco EXT ou DADOS.
Se EXT => Vá para 3); Se DADOS => vá para 4);
Se não for => Mensagem de erro (irrecuperável)

3. Se for EXT => Testa se já consta na tabela.
Se não constar na tabela, então guardar na tabela de subrotinas o nome da subrotina chamada, reservando mais 6 palavras para montar a subrotina chamadora.

Observação:

A solução adotada como mecanismo de ligação é bastante simplista: Quando não se dispõe de um endereço conhecido para o símbolo externo, cria-se um endereço conhecido na tabela de subrotinas, e usa-se este no lugar do endereço desconhecido.

Quando se tornar conhecido este endereço, inclui-se na tabela de subrotinas uma instrução de chamada para a subrotina propriamente dita, na área artificialmente criada para essa operação.

O resultado é a execução de algumas instruções a mais, tornando mais lento o processo. Dificulta-se também o mecanismo de passagem de parâmetros.

O formato dessa informação será como está mostrado na tabela a seguir:

Informações sobre chamadas externas

1 byte	Número da referência EXT	Para cada rotina, o número do EXT da mesma subrotina pode variar. Este byte indica qual é o número do EXT na subrotina ou no programa corrente.
2 bytes	Nome da subrotina	
2 bytes	X PLA *-*	Este endereço é calculável sabendo onde começa a tabela
2 bytes	PUG *-*	Reservado para PUG (Subrotina), que vai ser montado aqui quando a subrotina for carregada na memória (ou seja, quando for lido o correspondente bloco ENT na fita objeto relocável das subrotinas)
2 bytes	PLA X	Retorno ao programa chamador.

- Obs.: O “nome da subrotina” deverá estar fisicamente em outra área de memória para liberar o máximo espaço possível para o programa. Os outros seis bytes finais ficarão residentes na memória durante a execução do programa.
- Cada subrotina exige, para o link, de 6 bytes na tabela de subrotinas, e mais 3 bytes para uso do ligador-relocador (residente na memória durante a execução do programa, inclusive)
- Lido um bloco de DADOS, não pode aparecer nenhum novo EXT até que um bloco FIM seja detectado, e que um novo bloco de NOME seja eventualmente lido.

4. Se o bloco lido for um bloco de DADOS, o ligador-relocador só poderá aceitar em seguida outro bloco de DADOS ou um bloco de FIM, desde que o comprimento não exceda o que foi declarado no bloco de NOME, caso em que deverá se emitida uma mensagem de erro (*).
5. Processamento do bloco de DADOS:
 - (a) Somando ($I_1 I_2$) à constante de relocação, teremos o endereço absoluto onde deverá ser armazenado o primeiro dado (guardar este endereço em CI (contador de instruções))
 - (b) obter H_i para saber se o material que vem em seguida é absoluto ou relocável.
 - Se for absoluto => guardar J_i no endereço apontado em CI, e somar 1 em CI. Ignorar K_i neste caso.
 - Se for relocável => somar (J_i, K_i) com a constante de relocação. Testar se o código foi alterado (em caso afirmativo => erro de overflow de memória; caso contrário, guardar o resultado em (CI) e (CI)+1, e somar 2 em CI.
 - Se for relocável em relação à tabela de subrotinas => gerar link correspondente.
 - (c) Se não terminaram os dados do bloco, voltar a (b);
se terminaram, ler novo bloco.
- Se for bloco de DADOS, voltar a (a), caso contrário, deverá ser FIM (5)
6. Se o bloco encontrado for um bloco de FIM, o ligador-relocador deverá somar CR a (L_1, L_2) e montar um PLA [$(L_1, L_2)+CR$] na posição conveniente de memória para que o processamento se desvie para o programa no momento da execução.

(*) Todos os erros citados são irrecuperáveis, salvo referência em contrário.

Possivelmente o PLA será na posição 10 ($0A_{16}$) de memória e será gerado um bloco do código de saída contendo apenas este PLA.

Como a tabela de subrotinas será residente em memória ela também deverá ser gerada em fita absoluta como saída do ligador-relocador. (obs.: compilado o programa principal, o PUG (subrotina) ainda está indeterminado. ~~Na fita, é necessário prever um espaço para este PUG, o qual será montado quando a subrotina em questão for alocada na memória~~)(a tabela completa deve ser gerada só no fim, quando nenhuma referência externa estiver mais indeterminada).

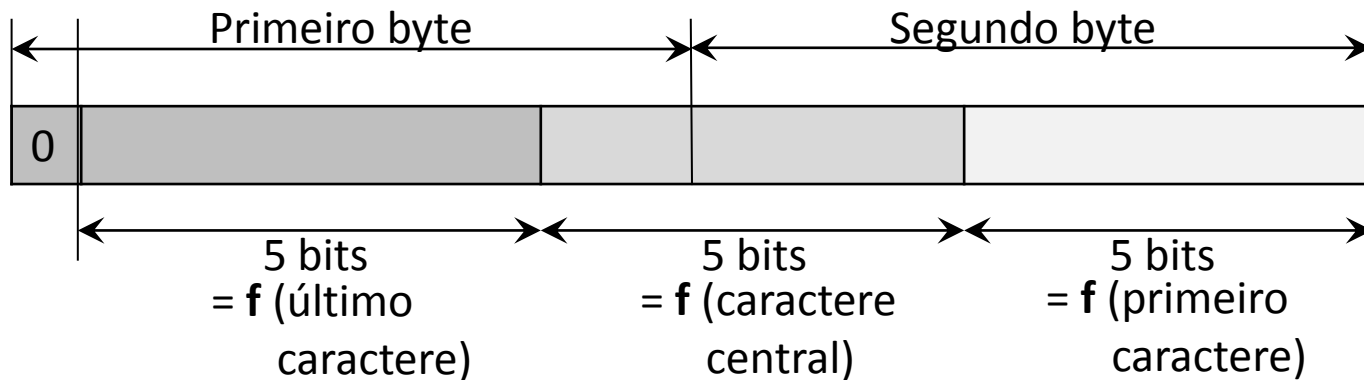
Outra coisa que o ligador-relocador deve fazer ao detectar um bloco de FIM: atualizar a constante de relocação ($CR \leftarrow CI$) e varrer a tabela de subrotinas para verificar se há ainda alguma subrotina indefinida.

Se ainda houver subrotinas indefinidas (o número de EXTs permanece menor que o número de ENTs correspondentes), voltar a ler nova fita (6), caso contrário, terminou o processo.

6. Ler um bloco. Testar se é NOME.
 - Se for => testar se é subrotina
 - Se não for subrotina => mensagem de erro (não pode haver mais de um programa principal)
 - Se for subrotina, o próximo bloco deve ser ENT.Guardar o nome na tabela de nomes e o comprimento.
Fazer tudo como no item 1.
Se deu tudo certo, ler o próximo bloco (7).

7. Ler bloco.
 - Se não for ENT => erro (subrotina sem entry point)
 - Se for ENT => ok => Procurar na tabela de subrotinas se este ENT particular foi solicitado e está indefinido.
 - Se ele estiver indefinido na tabela, o ligador-relocador deve montar o PUG (subrotina) do item 3, e na primeira palavra do NOME da SUBROTINA ligar o bit de “definido”.

Obs.: a rotina de empacotamento deixa o nome da subrotina com o seguinte formato:



O bit mais significativo da primeiro byte é sempre zero, e indica símbolo indefinido. Para defini-lo basta transformá-lo em “1”.

Caso o símbolo já esteja definido, [e só ler o restante da fita, testando da mesma maneira os ENTs (*).

(Obs.: se nenhum dos ENTs da subrotina que está sendo carregada tiver sido chamado, o ligador-relocador deverá subtrair novamente o comprimento do dados do CR1 e não deverá emitir código absoluto.

Se algum ENT for chamado, o CR1 deve ser mantido como estiver, e os dados deverão ser relocados, e o código absoluto correspondente deve ser gerado.

(*) Se algum bloco diferente de ENT for encontrado, vale a observação anterior.

Se houver geração de código porque houve uma chamada de ENT, deve-se testar o próximo bloco para verificar se é EXT.

Se for EXT, deve ser executado um procedimento idêntico ao descrito no item 3. (apenas se houver chamada de algum ENT da subrotina).

Terminados os EXTs, os blocos de dados devem ser tratados como em 4.

Um bloco de FIM deve apenas gerar a correção da constante de relocação ($CR1 \rightarrow CR$) e varrer a tabela de subrotinas para verificar se ainda há alguma indefinida.

Se houver alguma subrotina indefinida, voltar para 6.

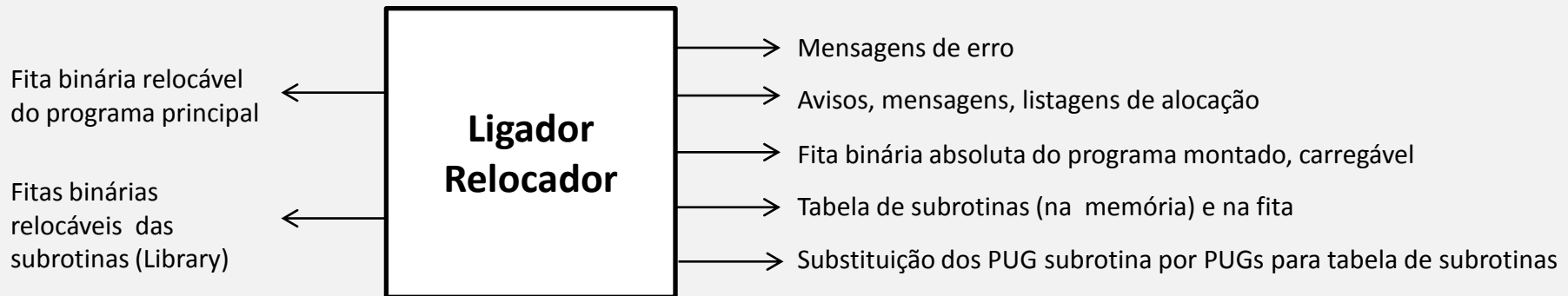
Caso contrário, terminou o processo. Deve desenrolar a fita da biblioteca, e parar só ao final da fita (20 bytes nulos seguidos).

ORGANIZAÇÃO DA MEMÓRIA E ESPECIFICAÇÕES GERAIS

Ligador-relocador – Organização da Memória

endereços	conteúdo
000 a 009	Reservadas para uso pelo carregador absoluto
00A a 00B	PLA programa
00C a 00F	outros dois ponteiros
010 a 0FF	Buffers de I/O, e áreas reservadas para transferências de parâmetros
100 a 1FF	Tabela de nomes de subrotinas, tag (in)definido e número da referência EXT
200 a ??? ??? a DFF	Área do ligador-relocador (programa, subrotinas e buffers). Fronteira variável, protegida por software. A partir da fronteira, tabela de subrotinas
E00 a F7F	Programas de utilidade geral
F80 a FFF	Loader absoluto - Fronteira protegida por hardware.

O que o Ligador-relocador deve fazer



- A ordem de entrada dos programas deve ser a seguinte: 1. Programa principal; 2. Subrotinas do usuário; 3. Subrotinas do sistema (library).
- O ligador-relocador deve, para economizar memória, ler bloco a bloco dos programas em formato relocável, gerar novos itens da tabela de subrotinas à medida que forem surgindo novos EXTs, atualizar as definições dos símbolos à medida que forem surgindo novos ENTs, e, durante todo esse processo, relocar o programa em questão, gerando em paralelo os links necessários e perfurando em fita o programa absoluto resultante em formato carregável pelo loader absoluto.
- Deve também fornecer códigos de erro e listagens de alocação à medida que uma nova subrotina ou programa viole qualquer das regras do jogo, ou termine a fita correspondente, respectivamente.

EXEMPLO ILUSTRATIVO DA OPERAÇÃO DO LIGADOR-RELOCADOR

Programa “fonte” (principal)

NOME	PRINCIPAL	(ENDEREÇO)	(CÓDIGO)
EXT	S1.		
EXT	S2.		
EXT	S3.		
EXEC PUG	S3.	000(R)	F002(X)
EXEC1 PUG	S2.	002(R)	F001(X)
CARI	/03	004(R)	DA03(A)
PUG	S1.	006(R)	F000(X)
PLA	EXEC	008(R)	0000(R)
FIM	EXEC1	002(R)	

Programa “fonte” (principal)

INÍCIO DA FITA	40 0U MAIS bytes nulos
BLOCO DE NOME	-1,PRL,0A BYTES,PROG.PRINCIPAL
SEPARAÇÃO	4 bytes nulos
BLOCO EXT	-3,S1.,00
SEPARAÇÃO	4 bytes nulos
BLOCO EXT	-3,S2.,01
SEPARAÇÃO	4 bytes nulos
BLOCO EXT	-3,S3.,02
SEPARAÇÃO	4 bytes nulos
BLOCO DADOS	-4,0000,-1,F002,-1,F001
SEPARAÇÃO	4 bytes nulos
BLOCO DADOS	0,DA00,0,0300,-1,F000
SEPARAÇÃO	4 bytes nulos
BLOCO DADOS	1,0000
SEPARAÇÃO	4 bytes nulos
BLOCO FIM	-5,0002
FINAL DA FITA	40 0U MAIS bytes nulos

SUBROTINA1,2

	NOME	SUBROTINA1,2		
	ENT	S1.		
	ENT	S2.		
	EXT	S3.	(ENDEREÇO)	(CÓDIGO)
S1.	PLA	*_*	000(R)	0000(A)
	UM		002(R)	81 (A)
	PLA	S1.	003(R)	0000(R)
S2.	PLA	*_*	005(R)	0000(A)
	PUG	S3.	007(R)	F000(R)
	SOMI	/02	009(R)	D802(A)
	PLA	S2.	008(R)	0005(R)
	FIM	0	000(R)	

SUBROTINA1,2

INÍCIO DA FITA	40 OU MAIS bytes nulos
BLOCO DE NOME	-1,SU2,0D BYTES,SUBROTINA
SEPARAÇÃO	4 bytes nulos
BLOCO ENT	-2,S1.,0000
SEPARAÇÃO	4 bytes nulos
BLOCO ENT	-2,S2.,0005
SEPARAÇÃO	4 bytes nulos
BLOCO EXT	-3,S3.,00
SEPARAÇÃO	4 bytes nulos
BLOCO DADOS	-4,0000,0,0000,0,0000,0,8100,1,0000,0,0000,0,0000
SEPARAÇÃO	4 bytes nulos
BLOCO DADOS	1,F000,0,D800,0,0200,1,0005
SEPARAÇÃO	4 bytes nulos
BLOCO FIM	-5,0000
FINAL DA FITA	40 OU MAIS bytes nulos

SUBROTINA3

NOME		SUBROTINA3		
ENT	S3.		(ENDEREÇO)	(CÓDIGO)
EXT	S4.			
S3. PLA	*_*		000(R)	0000(A)
PUG	S4.		002(R)	F000(X)
PLA	S3.		004(R)	0000(R)
FIM	0		000(R)	

SUBROTINA3

INÍCIO DA FITA	40 OU MAIS bytes nulos
BLOCO DE NOME	-1,SU3,06 BYTES,SUBROTINA
SEPARAÇÃO	4 bytes nulos
BLOCO ENT	-2,S3.,0000
SEPARAÇÃO	4 bytes nulos
BLOCO EXT	-3,S4.,00
SEPARAÇÃO	4 bytes nulos
BLOCO DADOS	-4,0000,0,0000,0,0000,-1,F000,1,0000
SEPARAÇÃO	4 bytes nulos
BLOCO FIM	-5,0000
FINAL DA FITA	40 OU MAIS bytes nulos

SUBROTINA4

NOME		SUBROTINA4		
ENT			(ENDEREÇO)	(CÓDIGO)
S4.	PLA	*_*	000(R)	0000(A)
	CARI	/05	002(R)	DA05(A)
	PLA	S4.	004(R)	0000(R)
	FIM	0	000(R)	

SUBROTINA4

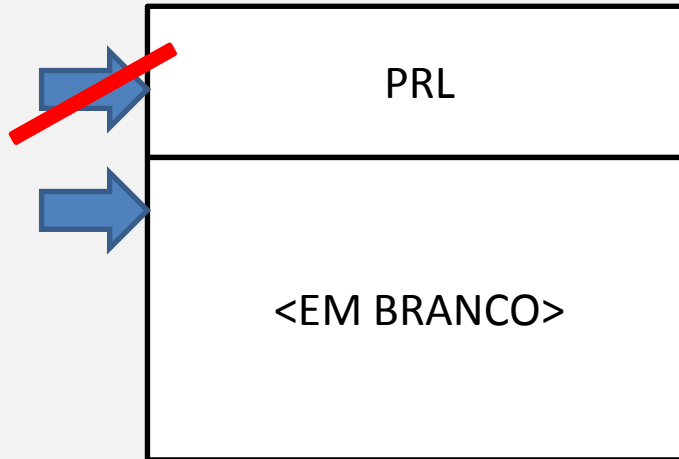
INÍCIO DA FITA	40 OU MAIS bytes nulos
BLOCO DE NOME	-1,SU4,06 BYTES,SUBROTINA
SEPARAÇÃO	4 bytes nulos
BLOCO ENT	-2,S4.,0000
SEPARAÇÃO	4 bytes nulos
BLOCO EXT	-3,S4.,00
SEPARAÇÃO	4 bytes nulos
BLOCO DADOS	-4,0000,0,0000,0,0000,0,DA00,0,0500,1,0000
SEPARAÇÃO	4 bytes nulos
BLOCO FIM	-5,0000
FINAL DA FITA	40 OU MAIS bytes nulos

OPERAÇÃO PASSO A PASSO DO LIGADOR-RELOCADOR E SUAS AÇÕES CORRESPONDENTES

- Os slides seguintes mostram passo a passo a operação do ligador-relocador para o exemplo que foi apresentado, proporcionando um material suplementar, cujo acompanhamento auxilia a entender a lógica desse programa de software básico e a depurar sua implementação.

1. Ler 1º bloco do programa principal

Tabela de nomes:



(devidamente inicializados)

, CR=10, CR1=10

, CR=10, CR1=1A

(1A = 10+0A)

Como 1A < limite da memória,
então pode-se continuar

2. Ler três blocos de EXT

DA REF. EXT.
NOME DA REF.
DEF / INDEF.

0	S1.	I
1	S2.	I
2	S3.	I
-1	<EM BRANCO>	
	<EM BRANCO>	

Tabela de nomes de subrotinas

/DEE > LIMITE DA FRONTEIRA PROTEGIDA,
LOGO PODE-SE PROSSEGUIR.

Z PLA *-* PUG *-* PLA Z	DEE-DEF DF0-DF1 DF2-DF3
Y PLA *-* PUG *-* PLA Y	DF4-DF5 DF6-DF7 DF8-DF9
X PLA *-* PUG *-* PLA X	DFA-DFB DFC-DFD DFE-DFE

Tabela de subrotinas

3. Ler bloco de DADOS

```
40 bytes nulos (INÍCIO DO BLOCO)
NÚMERO DE BYTES DO BLOCO
ENDEREÇO INICIAL 0010 = CR + (0000 = END. REL. DO BLOCO DE DADOS)
FD, EE
FD, F4
DA, 03
FD, FA
00, 10
CHECKSUM
04 bytes nulos (SEPARADOR - FINAL DO BLOCO)
```

4. Ler bloco de FIM

04 bytes nulos (SEPARAÇÃO DO BLOCO ANTERIOR)
NÚMERO DE BYTES DO BLOCO
ENDEREÇO DE EXECUÇÃO (LOADER ABSOLUTO) 000A
PLA EXEC1 0012 (ESTE PLA É ARMAZENADO NA POSIÇÃO /00A E APONTA A PRIMEIRA INSTRUÇÃO EXECUTÁVEL DO PROGRAMA CORRENTE)
CHECKSUM
40 bytes nulos (FIM DA FITA)

AÇÃO:

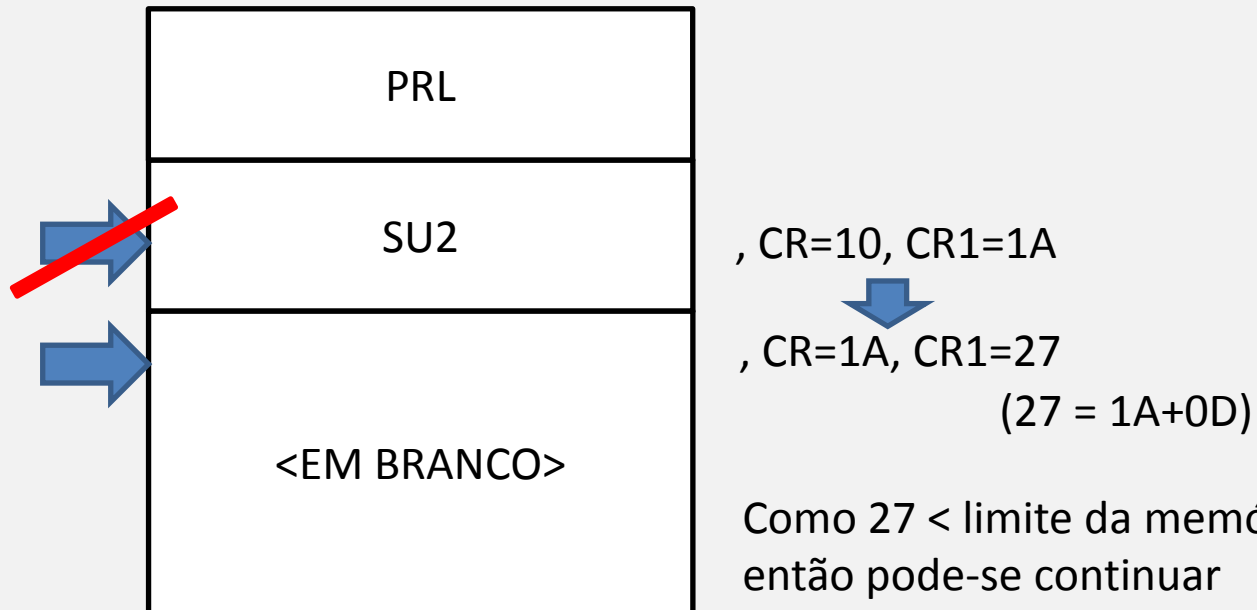
ATUALIZA $CR \leftarrow 1A$; $CR1=1A$.
EXISTEM TRÊS LINKS INDEFINIDOS,
PORTANTO DEVE CONTINUAR,
E PARA ISSO SOLICITA A LEITURA DE MAIS FITAS.

5. Acabou o código do programa principal

Havendo externals não resolvidos, e não sendo fornecidos códigos-objeto adicionais, encerra-se a execução do ligador-relocador reportando a ocorrência de um erro.

6. Ler primeiro bloco da subrotina 1,2

Tabela de nomes:



7. Ler dois blocos de ENT

DA REF. EXT.
NOME DA REF.
DEF / INDEF.

0	S1.	D
1	S2.	D
2	S3.	I
-1	<EM BRANCO>	
	<EM BRANCO>	

Tabela de nomes de subrotinas

CR = /01A; /01A = CR + /000; /01F = CR + /005

S3.

S2.

S1.

Z PLA *-* PUG /01A PLA /0FA	DEE-DEF DF0-DF1 DF2-DF3
Y PLA *-* PUG /01F PLA /DFA	DF4-DF5 DF6-DF7 DF8-DF9
X PLA *-* PUG *-* PLA X	DFA-DFB DFC-DFD DFE-DFE

Tabela de subrotinas

8. Ler o bloco de EXT

# DA REF. EXT.	NOME DA REF.	DEF / INDEF.
0	S1.	D
0	S2.	D
1	S3.	I
-1	<EM BRANCO>	
-1	<EM BRANCO>	

Tabela de nomes de subrotinas

S3.

S2.

S1.

Z PLA *-* PUG /01A PLA /0FA	DEE-DEF DF0-DF1 DF2-DF3
Y PLA *-* PUG /01F PLA /DFA	DF4-DF5 DF6-DF7 DF8-DF9
X PLA *-* PUG *-* PLA X	DFA-DFB DFC-DFD DFE-DFE

Tabela de subrotinas

9. Ler bloco de DADOS

```
4 bytes nulos (INÍCIO DO BLOCO)
NÚMERO DE BYTES DO BLOCO
ENDEREÇO INICIAL 001A = CR + (0000 = END. REL. DO BLOCO DE DADOS)
00, 00
81
00, 1A
00, 00
F0, 1A
D8, 02
00, 1F
CHECKSUM
04 bytes nulos (SEPARADOR - FINAL DO BLOCO)
```

10. Ler bloco de FIM

A fita de saída permanece inalterada;

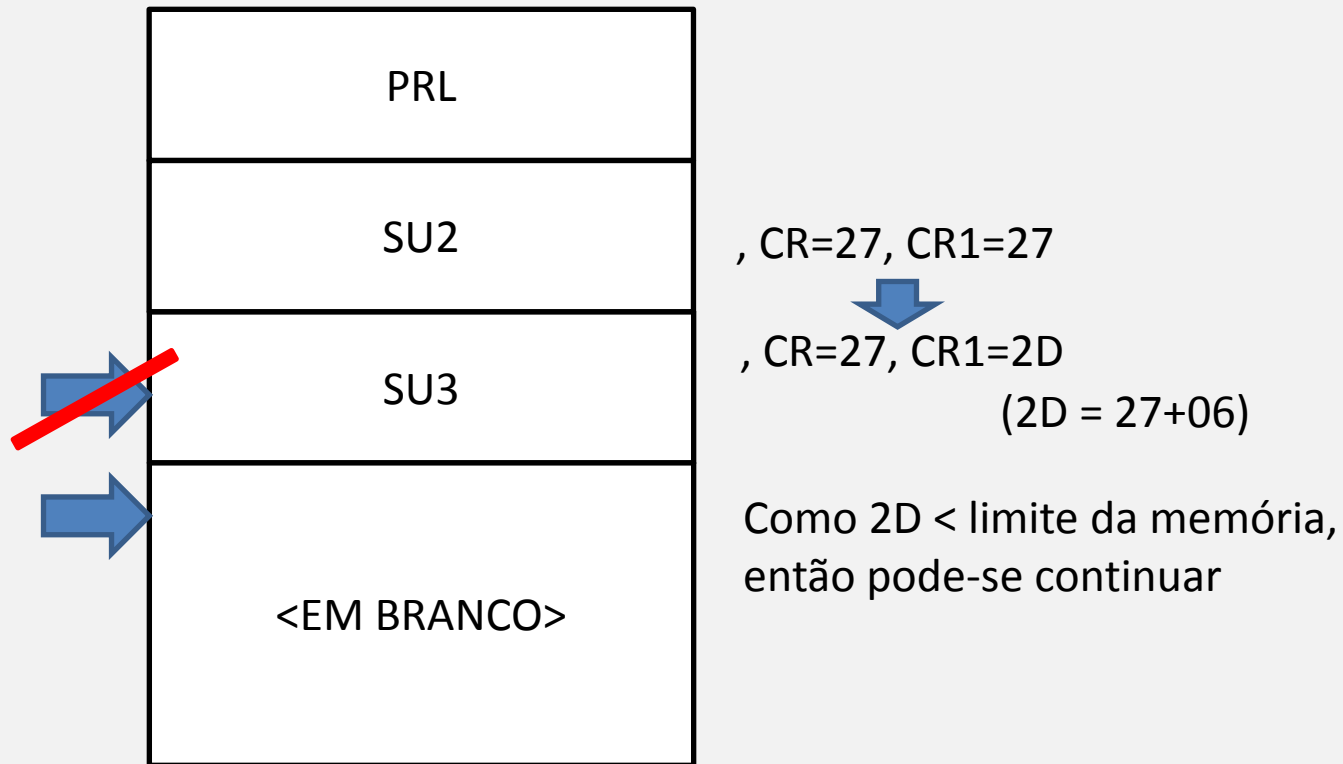
$CR \leftarrow 27$; $CR1 \leftarrow 27$;

Existe um link marcado como indefinido, portanto é preciso prosseguir:

Solicita a leitura de mais códigos relocáveis de entrada

11. Ler primeiro bloco da subrotina 3

Tabela de nomes:



12. Ler um bloco ENT, depois um EXT

DA REF. EXT.
NOME DA REF.
DEF / INDEF.

0	S1.	D
0	S2.	D
0	S3.	D
1	S4.	I
-1	<EM BRANCO>	

Tabela de nomes de subrotinas

CR = /027 ; /027 = CR + /000;

S4.

S3.

S2.

S1.

T PLA *-* PUG *-* PLA /DE8	DE8-DE9 DEA-DEB DEC-DED
Z PLA ** PUG /027 PLA /DEE	DEE-DEF DF0-DF1 DF2-DF3
Y PLA *-* PUG /01F PLA /DF4	DF4-DF5 DF6-DF7 DF8-DF9
X PLA *-* PUG /01A PLA /DFA	DFA-DFB DFC-DFD DFE-DFE

Tabela de subrotinas

/DE8 é maior que o limite protegido, portanto está tudo OK. Pode-se prosseguir.

13. Ler bloco de DADOS

4 bytes nulos (INÍCIO DO BLOCO)

NÚMERO DE BYTES DO BLOCO

ENDEREÇO INICIAL 0027 = CR + (0000 = END. REL. DO BLOCO DE DADOS)

00, 00

FD, E8

00, 27

CHECKSUM

04 bytes nulos (SEPARADOR - FINAL DO BLOCO)

14. Ler bloco de FIM

A fita de saída permanece inalterada;



$CR \leftarrow 2D$; $CR1 \leftarrow 2D$;

Existe um link marcado como indefinido, portanto é preciso prosseguir:

Solicita a leitura de mais códigos relocáveis de entrada

15. Ler primeiro bloco da subrotina 4

Tabela de nomes:

	PRL
	SU2
	SU3
	SU4
	<EM BRANCO>

, CR=2D, CR1=2D



, CR=2D, CR1=33

(33 = 2D+06)

Como $2D < \text{limite da memória}$,
então pode-se continuar

16. Ler bloco ENT, depois EXT

# DA REF. EXT.	NOME DA REF.	DEF / INDEF.
0	S1.	D
0	S2.	D
0	S3.	D
1	S4.	D
-1	<EM BRANCO>	

Tabela de nomes de subrotinas

/DE8 é maior que o limite protegido, portanto está tudo OK. Pode-se prosseguir.

CR = /02D ; /02D = CR + /000;

T PLA *-* PUG /02D PLA /DE8	DE8-DE9 DEA-DEB DEC-DED
Z PLA *-* PUG /027 PLA /DEE	DEE-DEF DF0-DF1 DF2-DF3
Y PLA *-* PUG /01F PLA /DF4	DF4-DF5 DF6-DF7 DF8-DF9
X PLA *-* PUG /01A PLA /DFA	DFA-DFB DFC-DFD DFE-DFE

Tabela de subrotinas

17. Ler bloco de DADOS

```
4 bytes nulos (INÍCIO DO BLOCO)
NÚMERO DE BYTES DO BLOCO
ENDEREÇO INICIAL 002D = CR + (0000 = END. REL. DO BLOCO DE DADOS)
00, 00
DA, 05
00, 2D
CHECKSUM
04 bytes nulos (SEPARADOR - FINAL DO BLOCO)
```

18. Ler bloco de FIM

CR \leftarrow 33; CR1 \leftarrow 33;

Não há mais links marcados como indefinidos, portanto o programa não vai solicitar a entrada de mais códigos relocáveis. Desenrola a fita de biblioteca que estiver na leitora, até encontrar 20 bytes nulos consecutivos, e encerra a leitura. Perfura então a tabela de subrotinas em formato carregável, e mais 40 bytes nulos para encerrar a fita absoluta gerada.

4 bytes nulos (INÍCIO DO BLOCO)

NÚMERO DE BYTES DO BLOCO

ENDEREÇO INICIAL 0DE8 endereço absoluto da tabela de subrotinas

00, 00, FO, 2D, 0D, E8, 00, 00, FD, 27, 0D, EE, 00, 00, F0, 1F,
0D, F4, 00, 00, F0, 1^a, 0D, FA

CHECKSUM

40 bytes nulos (SEPARADOR - FINAL DA FITA)

Observação

Incluir a geração do mapa da memória quando da carga de cada fita (no bloco de FIM)

	Nome (3 car.)	Começo (hexa)	Final (hexa)
NOM	□xxx	□xxx	
xxx	□xxx	□xxx	
xxx	□xxx	□xxx	

programa principal
 uma linha em branco
 (não pular linha...
 ...se for subrotina)

PSEUDO-CÓDIGO DO LIGADOR-RELOCADOR

Pseudo-código do ligador-relocador

- Por questões de editoração, os diagramas de blocos da documentação original foram transcritos para o formato de pseudo-código, apenas incorporando pequenas modificações para adequação ao formato desta apresentação.

tratamento do programa principal _{0,X,A}

0: Inicializar ponteiros e parâmetros

X: Ler um bloco (LEBLOC)

É um bloco de NOME de programa principal?

Se não, Erro (primeiro bloco não é de nome)

Pare. Após partida, voltar para X.

Se sim, ligar indicador de programa principal,

Atualizar tabela de NOMES,

Gerar 40 bytes nulos na TTY de perfuração,

Atualizar CR1,

Deu overflow de memória?

Se sim, Erro (overflow de memória)

Pare. Após partida, voltar para 0.

Se não, **A:** Ler um bloco (LEBLOC)

Desviar conforme o tipo:

EXT para 2;

DADOS para 3;

FIM para 4;

outros : Erro (bloco incorreto).

Pare. Após partida, voltar para A.

tratamento do programa principal 1,2,5,18,C

1: Erro (bloco fora de sequência)

Pare. Após partida, retornar para 0.

2: Iniciar contador de EXTs com 1

Colocar -1 em todas as referências EXT da tabela [e atualizar o ponteiro]

5: [Procurar o EXT na tabela de EXTs]

Procurar o EXT na tabela de EXTs

Achou?

Se sim, C: colocar o contador de EXTs na tabela de referências EXT
[e gerar o correspondente endereço]

Ler mais um bloco (LEBLOC)

Desviar conforme o tipo:

EXT: incrementar contador de EXTs e voltar para 5;

DADOS: para 3;

FIM: para 4;

outros: para 1.

Se não achou, colocar o novo EXT indefinido na tabela de EXTs

Gerar o endereço correspondente na tabela de subrotinas

Deu overflow na tabela de subrotinas?

Se não, atualizar ponteiro da tabela de subrotinas,
atualizar ponteiro de EXTs,
e voltar para C

Se sim, 18: Erro (overflow na tabela de subrotinas)
Pare. Após partida, voltar para 0

tratamento do programa principal 3,6,11,9,30,D

3: Somar (I_1, I_2) com CR e guardar na posição do endereço absoluto no buffer de saída

Iniciar contador de palavras do código objeto com 0 e ponteiro do buffer de saída

Calcular quantas “instruções” existem no bloco $I=N/3$

Guardar em um contador de instruções do bloco (CIB)

D : Carregar o próximo H_i

Desviar, conforme o tipo do bloco:

ABSOLUTO: para 6;

RELOCÁVEL: para 7;

EXTERNO: para 8;

outros: Erro (bloco incorreto).

Pare. Após partida, voltar para 0.

6: Montar no buffer de saída, logo depois do último dado, o conteúdo de J_i

Deslocar o ponteiro do buffer de saída

11: Decrementar CIB.

Se deu diferente de zero, deslocar o ponteiro para o próximo, e voltar para D

Se deu zero, terminou o bloco.

Completar com CHECKSUM e número de bytes

DUMPAR o bloco objeto gerado, em formato carregável.

9: Ler um novo bloco

Desviar, conforme o tipo do bloco:

DADOS: para 3;

FIM: 30: imprimir nome, início e fim do código (CR,CR1);

É final de subrotina?

Se sim, é subrotina, então desviar para 17

Se não, é programa principal, então voltar para 4.

outros: para 1;

tratamento do programa principal 4,7,8,12,17

- 4: Montar bloco de saída com: endereço 00A, um PLA para o endereço de execução relocado, e checksum.
Perfurar 4 bytes nulos.
- 17: Fazer $CR \leftarrow CR1$
[imprimir o mapa de memória]
Restam links indefinidos?
Se sim, voltar para 10
Se não, terminou a tarefa do ligador-relocador
Gerar o código relativo à tabela das subrotinas
Pare. Ao acionar partida, voltar para 0
- 7: Carregar (J_i, K_i) e somar com a constante de relocação
Deu overflow de memória (mudou o código)?
Se sim, Erro (Overflow de memória)
Pare. Após partida, voltar para 0.
Se não, montar o resultado da relocação nos dois bytes
que seguem o último dado, no buffer de dados de saída;
- 12: Somar 2 ao ponteiro do buffer de saída
Desviar para 11.
- 8: Carregar (J_i, K_i)
Separar a referência externa
Pesquisar na tabela de referências EXT
Encontrou?
Se não, Erro (símbolo declarado externo não consta na tabela de EXTs)
Pare. Após partida, voltar para 0.
Se sim, Calcular o endereço da subrotina correspondente, montada na tabela de subrotinas
Montar instrução com este endereço no buffer de saída
Desviar para 12

tratamento de subrotina 10-13-E-F-G-H-I

10: Imprimir mensagem solicitando mais fitas.

Pare. Após partida, prosseguir em E.

E: Ler um bloco

É bloco de NOME?

Se não, Erro (Bloco lido não é de NOME)

Pare. Após partida, voltar para E

Se sim, prosseguir em 13

13: É programa principal?

Se sim, Erro (Mais de um programa principal);

Pare. Após partida, voltar para 0.

Se não, prosseguir em F.

F: Desligar o indicador de programa principal

Atualizar a tabela de nomes

Guardar o comprimento em COMPR

~~Atualizar CR1~~

Deu overflow de memória?

Se sim, Erro (não cabe na memória disponível)

Pare. Após partida, voltar para 0.

Se não, prosseguir em G.

G: Ler um bloco

É bloco ENT?

Se não, Erro (subrotina não tem entry point)

Pare. Após partida, voltar para 0.

Se sim, prosseguir em H.

H: [ENT=1]

Procurar o entry point na tabela de EXTs

Achou?

Se não, [Colocar na tabela de EXTs]

[Atualizar tabela de ENTs]

Ler um novo bloco

Se tipo = ENT, voltar para H

Se não, [ENT=0]

A subrotina é desnecessária

[Muda ponteiros de NOME,

EXT, e da tab. de subrotinas]

~~Subtrair COMPR de CR1~~

[Subtrair 2 do ponteiro da

tabela de NOMEs]

I: Ler bloco

É bloco de NOME?

Se não, voltar para I.

Se sim, voltar para 13

[vai para dentro da subrotina EBN]

Se sim, prossegue em 14.

tratamento de subrotina 14-J-K

14: O EXT encontrado na tabela está definido?
Se sim, esta subrotina já foi relocada
Desviar para J.
Se não, Torná-lo definido
Montar PUG na tabela de subrotinas
[colocar ENTs na tabela de ENT]
Prosseguir em J.

J: Ler um bloco
Desviar conforme o tipo:
NOME: para 1
EXT: para 15
DADOS: para ~~16~~ 3
FIM: para 17
ENT: para K
outros: para 1

K: Procurar o entry point na tabela dos EXTs
Achou?
Se sim, voltar para 14
Se não, colocar como definido na tabela
voltar para J.

tratamento de subrotina 15-16-[19]-L-M

- 15: Inicializar contador de EXTs = 1
Colocar -1 em todas as referências EXT da tabela
- 16: [19]: Procurar o símbolo EXT na tabela de EXTs. Achou?
Se sim: M: Colocar o contador de EXTs na
tabela de referências EXT.
Prosseguir em L.
Se não: Gerar novo EXT não definido
Gerar mais uma subrotina
[endereço] na tabela de subrotinas
Deu overflow na tabela de subrotinas [endereço]?
Se sim: voltar para 18
Se não: Atualizar ponteiros e voltar para M.
- L: Ler um bloco
Desviar conforme o tipo:
EXT: incrementar o contador de EXTs e voltar para 16 [19]
DADOS: para 3
~~FIM: perfurar 4 bytes nulos e voltar para 17 (?)~~
Outros: para 1.