一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
SS2013-25, KBSE2013-25 (2013-07)

Malbolgeのワード長の拡大とそのプログラミング支援ツール

加藤 起騎† 酒井 正彦†† 坂部 俊樹†† 草刈 圭一朗†† 西田 直樹††

†,†† 名古屋大学 大学院情報科学研究科
〒464-8603 愛知県名古屋市千種区不老町
E-mail: †tatsuki@sakabe.i.is.nagoya-u.ac.jp, ††{sakai,sakabe,kusakari,nishida}@is.nagoya-u.ac.jp

あらまし Malbolgeは最も難解なプログラミング言語として知られている．近年，Malbolgeのための中間言語として低級アセンブリ言語が設計され，そのプログラムからMalbolgeプログラムを生成する低級アセンブラが構築された．しかし，低級アセンブリ言語を用いてプログラミングを行う際，メモリ不足という事態が度々発生していた．例えば，低級アセンブラを利用した数値のインクリメントを行うMalbolgeプログラム生成は，それだけでメモリ空間59049ワードのうち10分の1も消費する．本稿では，この問題の解決のためにMalbolgeのワード長を10tritsから20tritsに拡大し，3^{20}ワードのメモリを持つMalbolge20を提案する．Malbolge20では，3^{20}ワードという膨大の量のメモリを扱うため，メモリの管理方法を大きく変更する．また，Malbolgeを対象としている低級アセンブラ及びMalbolgeデバッガをMalbolge20に対応させ，Malbolge20のプログラミング環境を整備する．

キーワード 難解プログラミング言語，Malbolge20，メモリ管理

Malbolge with 20trits word length and its programming

Tatsuki KATO†, Masahiko SAKAI††, Toshiki SAKABE††,

Keiichirou KUSAKARI††, and Naoki NISHIDA††

† Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya-City, Aichi, 464-8603 Japan
E-mail: †tatsuki@sakabe.i.is.nagoya-u.ac.jp, ††{sakai,sakabe,kusakari,nishida}@is.nagoya-u.ac.jp

Abstract Malbolge is known to be one of the most esoteric programming languages. Recently a low-level assembly language (LA-language) has been designed as an intermediate language for Malbolge programming and a low-level assembler (LA-assembler) has been constructed that generates a Malbolge program from a low-level assembly program. We have a problem that the LA-assembler often fails because the size of generated Malbolge program exceeds the limit. For example, the size of an incrementation program produced by the LA-assembler is one-tenth of the allowed size. In order to solve this problem, this paper proposes a variant of Malbolge, named Malbolge20, whose word length is extended to 20trits from the original size 10trits. We enhanced the memory management by introducing cash mechanism. We modify the existing LA-assembler and debugger of Malbolge for Malbolge20 as a programming environment of Malbolge20.

Key words Esoteric Programming Language, Malbolge20, memory management

1. はじめに

難解プログラミング言語は意図的にその言語でのプログラミングが困難になるように設計された言語である．このような言語で書かれたプログラムは解読困難性を持つため，情報セキュリティにおいてプログラムの改ざん防止や知的財産保護に役立つと考えられている．

Malbolge [1] は，数ある難解言語の中でも特に難解として知られており，プログラムの解析はもちろんのこと，プログラムの作成すら非常に困難である．しかし近年，飯澤らによりMalbolgeのための中間言語として低級アセンブリ言語が設計され，そのプログラムからMalbolgeプログラムを生成する低級アセンブラが構築された [6] [7]．また，長坂らによりMalbolgeデバッガが構築され，Malbolgeプログラミング環境の整備が進んでいる [8] [9]．低級アセンブリ言語には通常のプログラミング言語が持つような演算命令がなく，Malbolge

特有の演算を行う命令のみであるため，低級アセンブリ言語でのプログラミングには困難が伴う．これを解決するため，低級アセンブリプログラム作成のための高級アセンブリ言語が飯澤らによって構築され [6] [7] 安藤らによって使いやすさが改良されている [3] [4] [5]．すなわち既存研究では，高級言語のプログラムを高級アセンブリプログラム，低級アセンブリプログラム，Malbolge プログラムの順に変換していくことで，高級言語のプログラムを Malbolge プログラムに変換する．しかし，この実用化においてはメモリ不足という問題点がある．例えば，低級アセンブラを利用して作成した数値のインクリメントを行う Malbolge プログラムは，それだけでメモリ空間 59049 ワードのうち 10 分の 1 も消費する．

本稿では，この問題の解決のために Malbolge のワード長を 10trits から 20trits に拡大した Malbolge20 を提案する．また，3^{20} ワードのメモリを持つ Malbolge インタプリタを実装法ならびに，Malbolge プログラム生成を目的として開発された低級アセンブラと Malbolge デバッガを Malbolge20 に対応させる方法について述べる．

本稿の構成は次の通りである．2 節では，準備として Malbolge の仕様について説明する．3 節では，Malbolge20 の定義を定め，そのインタプリタの実装を行う．4 節で，これまで行われていなかった低級アセンブラの仕様の明確化を行い，5 節で低級アセンブラ，Malbolge デバッガを Malbolge20 に対応させる．6 節で全体のまとめを行う．

2. 準　　備

本節では，Malbolge の仕様と既存研究の成果の一つである Malbolge デバッガについて説明する．飯澤らの提案した Malbolge でのプログラミング手法については，文献 [6] [7] に詳細が示されている．

2.1 Malbolge

Malbolge は仮想マシン上で動く機械語であり，C 言語によって実装されたインタプリタ [2] によってその意味が定められている．仮想機械は 3 つのレジスタ (アキュムレータ A，コードポインタ C，データポインタ D) とメモリ (mem) を持ち，値は三進数十桁 (10trits) で表現される．よって値は 0000000000t～2222222222t となり，メモリのアドレス空間も mem[0]～mem[59048] で定義される．以降では，三進数表記を十進数と区別するため，末尾に t をつけて 0012t(=5) のように表記する．

インタプリタは，プログラムの実行前に，入力の Malbolge プログラムをメモリにロードする．ただし，入力となる Malbolge プログラムは，印字可能文字 (ASCII 値:33～126) の列であり，コード長は 59049($=3^{10}$) 文字未満でなければならない．ソースコード上の 1 文字がメモリ上の 1 ワードに対応し，スペースと改行は無視される．ただし，p 文字目における印字可能文字 c は表 1 の変換用配列 xlat1 による変換:xlat1[(c-33+p)%94] の値が i,j,p,*,/,<,v,o の 8 文字のいずれかに該当せねばならず，それ以外の場合はロード時エラーとなる．このように，Malbolge のメモリの初期値には著しい制約があり，プログラミングが困難である理由の一つになっている．

表 2　Malbolge の命令

xlat1 による値	表記	説明
i	JMP	ジャンプ．C:=mem[D].
j	MOV_D	D レジスタの更新．D:=mem[D].
p	OPR	演算命令．A,mem[D]:=op(A,mem[D]).
*	ROT	右ローテート．A,mem[D]:=rotr(mem[D]).
/	INPUT	入力．A:=getchar().
<	OUTPUT	出力．putchar(A).
v	HALT	終了．プログラムの実行を停止.
o	NOP	無操作．何も行わない.

表 3　op の各桁の演算

		X_i		
		0	1	2
Y_i	0	1	0	0
	1	1	0	2
	2	2	2	1

余ったメモリ領域については，後述する関数 op を用いて最後の 2 ワードからフィボナッチ数列のように，for(m=コードサイズ; m<59048; m++) [m]=op([m-1],[m-2]) で初期化する．

初期化の後は，その時点での C および [C] の値を元に，変換表 xlat1 による変換 : xlat1[([C]-33+C)%94] の値から表 2 に従って命令を決定し，順次 1 ステップずつ実行する．便宜上，これ以降で Malbolge の命令は表 2 のニーモニック表記で記述する．8 つの命令のうちいずれにも解釈できない場合は NOP と同じ動作となる．OPR 命令の演算 op(X,Y) は，三進数で表された二引数の各桁同士で表 3 に従って計算を行う．以下に op 演算の例を示す．

op(0120120120t, 0001112222t) = 1001022212t

ROT 命令の演算子 rotr は 10trits の値に対して以下のように右ローテートを行う．

rotr(0001112222t) = 2000111222t

命令を 1 ステップ実行する度に，図 2 の変換表 xlat2 を用いて [C] の値を xlat2[[C] - 33] に書き換える．この処理のため，あるコードを実行した時，実行後にそのコードは全く別の値に書き換えられてしまう．このため，Malbolge ではコードの反復実行が非常に困難になっている．

1 ステップ実行終了後，C と D をインクリメントして次の命令を実行する．C または D の値が最大値 59048 だった場合，インクリメントされると値は 0 となる．HALT 命令が実行されるまで実行は続けられる．

2.2 Malbolge デバッガ

Malbolge デバッガ [9] は，インタプリタの動作を GUI で表示する実行トレーサーである．その入力は Malbolge プログラムであり，以下の機能を持つ．

- 通常実行

ロードした Malbolge プログラムが終了するまで実行する．

- ステップ実行

指定された回数のステップを実行する．ステップ数は 1000 万回までの複数回指定できる．

表 1　変換表 xlat1

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
xlat1[i]	+	b	(	2	9	e	*	j	1	V	M	E	K	L	y	C	}	)	8	&	m	#	~	W	>	q	x	d	R	p	0	w	k	r	U	o	[	D	7	,	X	T	c	A	"	l	I
i	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93
xlat1[i]	.	v	%	{	q	J	h	4	G	¥	-	=	0	@	5	‘	_	3	i	<	?	Z	’	;	F	N	Q	u	Y	]	s	z	f	$	!	B	S	/	|	t	:	P	n	6	^	H	a

表 4　変換表 xlat2

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46
xlat2[i]	5	z	]	&	g	q	t	y	f	r	$	(	w	e	4	{	W	P	)	H	-	Z	n	,	[	%	¥	3	d	L	+	Q	;	>	U	!	p	J	S	7	2	F	h	0	A	1	C
i	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93
xlat2[i]	B	6	v	^	=	I	_	0	/	8	|	j	s	b	9	m	<	.	T	V	a	c	‘	u	Y	*	M	K	’	X	~	x	D	l	}	R	E	o	k	N	:	#	?	G	"	i	@

- ブレイクポイント設定

ブレイクポイント設定は，実行命令名，あるいはアドレスで指定する．命令指定は，指定された 1 つの命令が実行されるまで実行を繰り返す．アドレス指定は，複数のメモリアドレスにブレイクポイントを置くことができるようにし，指定された状態で実行すると，C，D レジスタのいずれかが指定アドレスと一致するまで実行を繰り返す．

- 仮想マシンの状況の表示

実行途中のメモリの中身とレジスタ値，それまでに実行したステップ数を表示する．メモリ値の表し方は，10 進数，3 進数，ASCII 文字，命令のニーモニック表記の 4 種類である．

- 実行履歴の表示

ステップ実行により実行された場合にはその実行の詳細を保存しておくことで，後から実行履歴を確認できる．

3. Malbolge のワード長の拡大

本節では，Malbolge のワード長を 10trits から 20trits に拡大し，3^{20} ワードのメモリを持つ Malbolge20 を提案する．ワード長を 20trits に拡大することで，メモリ空間はオリジナルの 3^{10} 倍である 3^{20} ワードとなる．

3.1 Malbolge20

Malbolge20 の意味は Malbolge と同様に仮想機械上で動作する機械語として定義される．Malbolge20 の仮想機械もメモリ (mem) と 3 つのレジスタ (A, C, D) を持つ．ただし，1 ワードは三進数二十桁 (20trits) であり，メモリアドレスも 20trits で表現される．値の範囲は，三進数表現で 00000000000000000000t～22222222222222222222t，十進数表現では 0 ～ $3^{20} - 1$ となる．また，Malbolge20 の命令は，Malbolge と同様 8 種類の命令を持つ．それらの意味は，10trits から 20trits への自然な拡張になっている．以下に，命令 OPR で行われる op 演算と，ROT 命令で行われる rotr 演算の具体例を示す．

op(01201201201201201201t, 00000000111111222222t)
= 10010010210210122122t

rotr(00000000111111222222t) = 20000000011111122222t

3.2 インタプリタの拡張

ここでは，Malbolge のワード長を 20trits 化させるためのインタプリタの拡張手法を示す．拡張の際の問題点は以下の通りである．

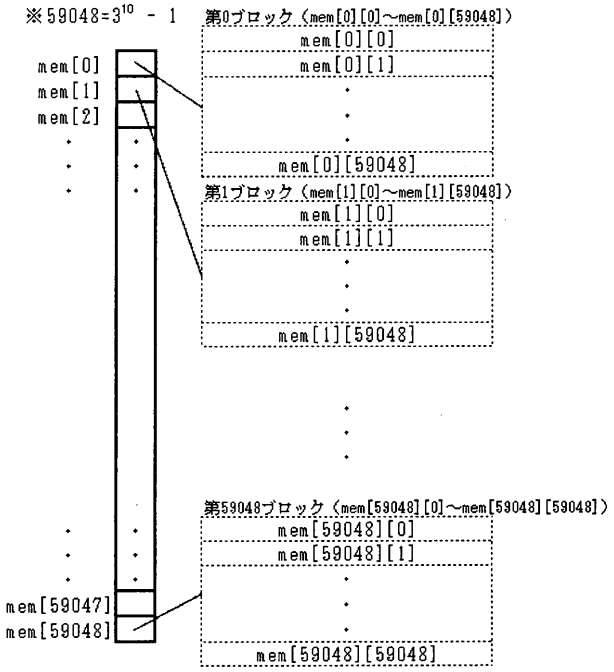

図 1　Malbolge20 のメモリイメージ図

- メモリのデータ構造
- メモリの初期化方法

メモリのデータ構造

Malbolge インタプリタではメモリを一つの配列で確保していた．これに対して Malbolge20 の 3^{20} ワードのメモリサイズは約 14GB もあり，一つの配列で確保するのは現実的ではない．このため，Malbolge20 インタプリタではメモリを 3^{10} 個にブロック化して管理する．1 つのブロックの大きさは 3^{10} ワードである．図 1 が，Malbolge20 のメモリ空間のイメージである．各ブロックは初めて利用される際に領域が確保され，後述する方法で初期化される．

メモリの初期化

2 節で紹介したように，Malbolge ではプログラムがロードされなかった未使用の領域は，直前の 2 ワードの値を元に次の値を順次計算していくフィボナッチ風の計算で初期化される．具体的には，[m]:=op([m-1],[m-2]) という処理を入力プログラムの最後のアドレスの次からメモリの終わりまで順に計算することでメモリ初期化を行う．例えば，メモリの n-2 番地の値

表 5　Malbolge の初期化

メモリ番地	メモリ値	計算式
mem[n-2]	0001112222t	
mem[n-1]	0120120122t	
mem[n]	1001022211t	op(mem[n-1], mem[n-2])
mem[n+1]	0120110222t	op(mem[n], mem[n-1])
mem[n+2]	1001022122t	op(mem[n+1], mem[n])
mem[n+3]	0120120211t	op(mem[n+2], mem[n+1])
mem[n+4]	1001012222t	op(mem[n+3], mem[n+2])
…	…	

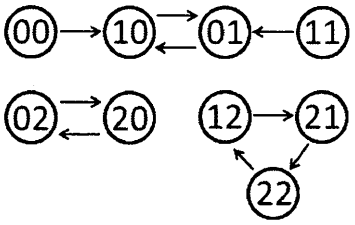

図 2　初期化処理における op への引数組み合わせの遷移

表 6　あるブロックの最初 2 ワードと次ブロックの最初 2 ワードの関係

第 n ブロックの最初 2 ワード	0	0	0	1	1	1	2	2	2
(mem[n][0], mem[n][1] の値)	0	1	2	0	1	2	0	1	2
第 n+1 ブロックの最初 2 ワード	0	1	2	0	1	1	0	2	2
(mem[n+1][0], mem[n+1][1] の値)	1	0	0	1	0	2	2	1	2

が 0001112222t，n-1 番地の値が 0120120122t であった場合，n 番地以降の値は表 5 のようになる．

Malbolge20 ではメモリを必要になったブロックから領域確保し初期化するため，飛び飛びにブロックが利用される場合に初期化が出来ない．そこで，Malbolge20 インタプリタの初期化を設計する前に，まず初期化で用いられる計算を解析する．初期化に用いる演算 op はトリット毎の演算でありワード長には本質的に関係しないため，Malbolge の初期化の説明で使用した表 5 を用いて説明する．表 5 における各メモリ値の最上位桁に注目すると，n 番地を計算するために，第一引数として mem[n-1] の最上位桁，第二引数として mem[n-2] の最上位桁が与えられる．即ち，op(0,0) の結果である 1 が mem[n] の最上位桁となる．他の桁も同様の計算が行われている．

このことから，各桁の初期化は直前の 2 ワードの該当桁の組み合わせを状態として持つオートマトンとして形式化出来る．オートマトンの状態"t_1t_2"は次番地のメモリ値を $op(t_1, t_2)$ によって計算する状態であることを意味する．従って，状態"t_1t_2"から遷移する状態は"$t_1\ op(t_1, t_2)$"となる．この考え方により得られるオートマトンを図 2 に示す．

遅延初期化処理

メモリ初期化計算の解析から，Malbolge20 の初期化では

(1)　予め各ブロックの最初の 2 ワードを計算しておく

(2)　個々のブロックの初期化は，その領域が確保された際に，(1) で計算しておいた 2 ワードの情報を元に行う

(1) に関して，あるブロックの最初 2 ワードから次ブロックの最初 2 ワードは図 2 の情報を利用することで効率よく計算できる．これをまとめたのが表 6 である．(2) に関して，実際にブロックの初期化が行われるのは，それらが初めて利用される際，即ち，レジスタが MOV_D 命令や JMP 命令によって未使用のブロックにジャンプした場合である．図 3 が JMP，MOV_D 命令が実行される時の処理の流れである．JMP，MOV_D 命令が実行される際，まず [D] の値から移動先であるブロック及びブロック内での番地を決定する．移動先のブロックがすでに存在する場合には，そのままレジスタを移動する．初期化されていない場合には，まずは該当ブロックの領域を確保し，初期化処理を行なってからレジスタを移動して終了する．

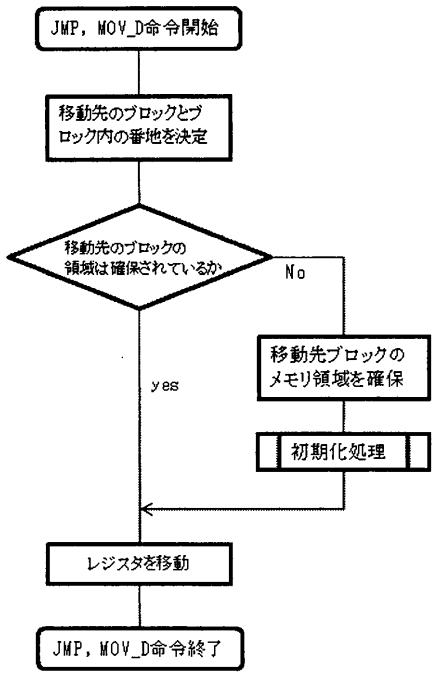

図 3　JMP, MOV_D 実行時の処理

4.　低級アセンブラの仕様

Malbolge プログラミングの支援ツールの一つとして，飯澤らにより低級アセンブラが開発されている．本節では低級アセンブラの処理概要を整理し，述べる．

低級アセンブラは，低級アセンブリプログラムを入力として，対応する Malbolge プログラムを出力する．処理は以下の順番で行われる．

(1)　低級アセンブリプログラムの読み込み

低級アセンブリプログラムを読み込み，アセンブリプログラムを動作させるのに必要なメモリの初期状態を記述したメモリイメージファイル (図 4) を作成する．メモリイメージファイルには，各メモリ番地に対応するメモリ値のみが順に記述されており，どのような値でも良いという場合には 0xFFFF が記述されている．しかし，Mlabolge のメモリ初期値の制約から，メモリイメージファイル内の多くのメモリ値は Malbolge 命令として記述出来ない．このため低級アセンブラは，実行によりメモリイメージファイル通りのメモリ状態を作成した後に，低級アセンブリプログラムの実行開始アドレスにジャンプするような Malbolge プログラム (図 5) を生成する．メモリイメージの

うち，Malbolge 命令として記述出来ない値に関しては，任意のメモリ値を指定されたメモリ番地に展開するコードを生成する方法 [6] [7] を用いて生成する．

以下の (2) と (3) は，メモリイメージファイル通りのメモリ状態を実行途中に作成する Malbolge プログラムの実現方法である．

(2) データモジュールの作成

図 5 における②の場所に，メモリ値の展開のために必要なデータモジュール [6] [7] を作成するためのコードを生成する．データモジュールのデータ列及びその作成方法は経験的に発見されたものであり，文献 [6] [7] にその詳細が示されている．

(3) 特定の値を設定するコードを作成

Malbolge 命令としては与えることが出来ない値 (図 4 の mem[l]～mem[m]) を作成するためのコードを③に生成する．このコードが実行されることで，目的のメモリ値がメモリ上に展開される．Malbolge 命令は置かれる位置によりその値が異なり，また特定の値の生成コードの長さはその値によって異なるため，図 5 の③の大きさや④の大きさを予測することは難しい．また，③の大きさは⑤によっても変化するため，⑤の位置，即ち l と m の値を決めるのは容易でない．このため現在はユーザが試行錯誤により決める必要がある．

(4) 命令化可能なコードを設定

図 5 における④の場所に直接 Mabolge プログラムとしてロード可能なコードを設定する．このコードは，メモリイメージファイルにおけるインタプリタがそのまま命令として解釈するメモリ値のことである．従って，図 4 における mem[j]～mem[k] の値を図 5④にコピーすることでこの処理は完了する．

(5) 初期化用末尾調整

Malbolge 特有の残りメモリの初期化処理が行われた際，生成された Malbolge プログラムの前提となる初期値，即ち，偶数番地が 81(0000010000t)，奇数番地が 29443(1111101111t) となるように末尾 2 文字を設定する．これは 81 がデータモジュール内にポインタを移動させる時に都合の良い値であり，なおかつ 81，29443 ともに 0000000000t を A レジスタに入れて OPR すればどちらも 1111111111t に初期化できる利点を持っているからである．

5. プログラミング支援ツールの 20trits 化

4 節までに紹介した Malbolge デバッガおよび低級アセンブラは Malbolge に対応して開発されたため，3 節で示した Malbolge20 に対して使用することはできない．そこで本節では，これらのツールの 20trits 化を行う．以降では区別するために，4 節までに紹介したツールは"10trits 用"を先頭につけて，本節で 20trits 化したツールは"20trits 用"を先頭につけて表記する．

5.1 低級アセンブラ

低級アセンブラの 20trits 化における大きな変更点は次の 2 点である．

- 低級アセンブリ言語の仕様
- 特定のメモリ値の作成方法

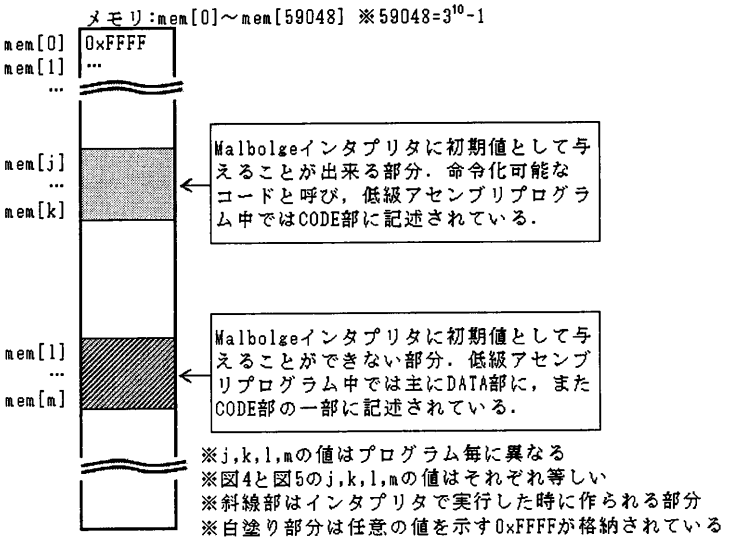

図 4 メモリイメージファイルの内容

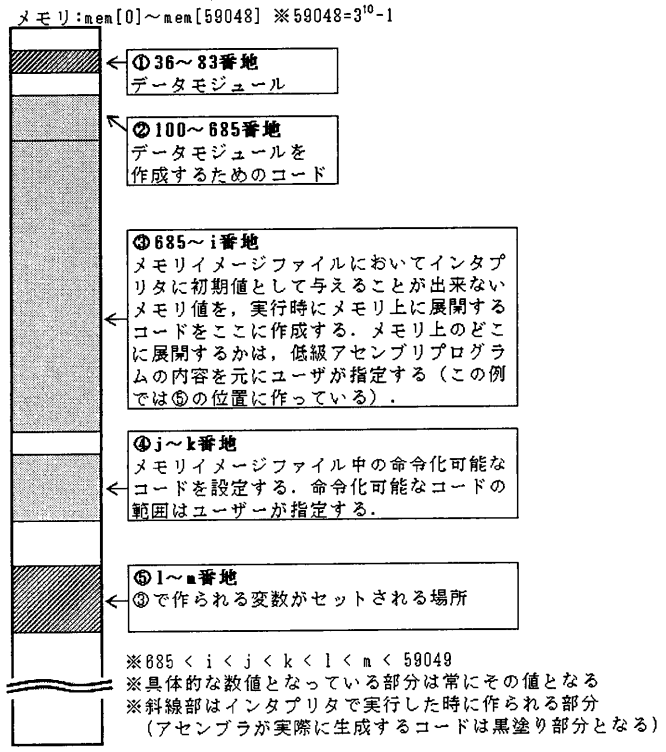

図 5 低級アセンブラによって作成される Malbolge プログラム

基本的な処理の流れは同じであるため，ここでは上記 2 点について述べる．

5.1.1 低級アセンブリ言語の仕様変更

入力の低級アセンブリプログラム内で数値を 10trits で表していた場合，20trits 表記に変更する必要がある．

例)

変更前		変更後
ROT 0000000012t	→	ROT 00000000000000000012t

また，実アドレス値は 0 から 59048 のいずれかとなっていたが，20trits 用のものでは 0 から 3486784400(= $3^{20}-1$) まで許される．

5.1.2 特定のメモリ値の作成方法

10trits 用低級アセンブラにおける特定のメモリ値の作成は，三進数十桁のメモリ値を状態とし，ROT と OPR による状態遷移を考え，その上で最短経路問題を解くことで行っていた．この手法によって生成される Malbolge コードは比較的短いが，三進数二十桁の値に適用すると最悪の場合で状態数が $3^{20}(> 2^{31})$ 個と爆発してしまい，実用的な時間で計算することが出来ないため，20trits 用低級アセンブラでは，1trit ずつ目的の値に設定していく手法 [6] [7] を用いる．この手法はでは，例えば 81(0000010000t) から 29524(1111111111t) を作成する場合は，$0000010000t \rightarrow 0000010001t \rightarrow 0000010011t \rightarrow 0000010111t \rightarrow \cdots \rightarrow 1111111111t$ というように，1trit の値の作成と左シフトを繰り返す．ただし，生成される Malbolge コードは，ほとんどの場合で冗長になることも分かっているので，更なる改良が望まれる．

5.2 Malbolge デバッガ

Malbolge デバッガの 20trits 化を行った際の大きな変更点は，新たに Malbolge20 のインタプリタを組み込んだ点である．新しいインタプリタの仕様に沿って，通常実行やステップ実行，メモリの表示などの基本的な機能を 10trits 用デバッガと矛盾がないように 20trits に対応させた．また，新しい機能として次の機能を追加した．

- ブロック単位でのメモリ表示

Malbolge20 では，メモリを二次元配列によりブロック化して管理している．このため，数値 n を指定して，第 n ブロック (mem[n][0]～mem[n][59048]) のメモリ状態を調べる機能を追加した．第 n ブロックの領域が確保されているか，また確保されている場合は詳しいメモリ状態を調べることができる．

6. ま と め

Malbolge のワード長を 10trits から 20trits に拡大し，それを Malbolge20 とした．次に，既存研究の成果である低級アセンブラの仕様を明確にした．また，同じく既存研究の成果である Malbolge デバッガ，低級アセンブラをそれぞれ Malbolge20 に対応させ，Malbolge20 のプログラミング環境を構築した．

10trits 用の低級アセンブラによる Malbolge プログラム生成がメモリ不足のため不可能であると判明した場合，本研究で作成した 20trits 用低級アセンブラによって Malbolge20 プログラムとして生成しなおせばよい．しかし，10trits 用として作成された低級アセンブリプログラムを本研究で構築した 20trits 用低級アセンブラに入力として与えるためには，低級アセンブリプログラムの記述を 20trits 用に修正する必要がある．研究の効率化のために，20trits 用低級アセンブラをこの修正を必要としないものへ改良することが今後の課題である．

文　献

[1] Ben Olmstead : “Malbolge: Programming from Hell”, http://www.bouletfermat.com/danny/malbolge/, 1998.

[2] Ben Olmstead:“Reference Malbolge Interpreter”, http://www.lscheffer.com/malbolge_interp.html.

[3] 安藤聡，酒井正彦，坂部俊樹，草刈圭一朗，西田直樹，Malbolge の高級アセンブリ言語への加算命令の追加，日本ソフトウェア科学会第 28 回大会講演論文集，No. 5A-3，2011.

[4] 安藤聡，酒井正彦，坂部俊樹，草刈圭一朗，西田直樹，Malbolge の高級アセンブリ言語への配列機能の追加，信学技報，電子情報通信学会，vol.112，No. 23，pp. 43--48，2012.

[5] 安藤聡，難解言語 Malbolge のプログラム作成のための中間言語の拡張と開発支援，名古屋大学修士論文，2013.

[6] 飯澤恒，坂部俊樹，酒井正彦，草刈圭一朗，西田直樹，難読プログラミング言語 Malbolge におけるプログラム構成手法，信学技報，電子情報通信学会，Vol.105，No. 129，pp. 25--30，2005.

[7] 飯澤恒，難解言語 Malbolge に基づくプログラム難読化に関する研究，名古屋大学修士論文，2006.

[8] 長坂哲，酒井正彦，坂部俊樹，草刈圭一朗，西田直樹，難解言語 Malbolge のチューリング完全性について，信学技報，電子情報通信学会，Vol. 110，No. 227，pp. 55--60，2010.

[9] 長坂哲，難解言語 Malbolge の弱チューリング完全性とプログラミング環境，名古屋大学修士論文，2011.