3.10 Code 128 - barcode generation

Write a program in the RISC-V assembly language which encodes a text using selected subset of Code 128 barcode.

Code 128 description

Code 128 encodes 128 symbols from one of three sets (Code Set A, Code Set B, Code Set C). The start symbol determines which subset is used. There are four widths of bars and spaces. Each symbol consists of three bars and three spaces (except stop symbol). Stop symbol consists of four bars and three spaces. The widths of bars and spaces are integer (1/2/3/4) multiples of the width of narrowest bar or space.

Code 128 consists of:

- quiet zone (empty space) – ten times the width of narrowest space or bar,
- start symbol,
- encoded data,
- check symbol,
- stop symbol,
- quiet zone.

Check symbol

The check symbol is calculated according to the formula:

$$check\_symbol\_value = \left(start\_symbol\_value + \sum_{i=1}^{data\ count} i * data\_value[i] \right) mod\ 103$$

Character encoding

The three Code 128 sets (Code Set A, Code Set B, Code Set C) are presented in table 1. Last column contains relative widths of bars (B) and spaces (S).

Table 1. Character codes

Value	Code set A	Code set B	Code set C	B S B S B S
0	SP	SP	00	2 1 2 2 2 2
1	!	!	01	2 2 2 1 2 2
2	"	"	02	2 2 2 2 2 1
3	#	#	03	1 2 1 2 2 3
4	$	$	04	1 2 1 3 2 2
5	%	%	05	1 3 1 2 2 2
6	&	&	06	1 2 2 2 1 3
7	'	'	07	1 2 2 3 1 2
8	(	(	08	1 3 2 2 1 2
9	)	)	09	2 2 1 2 1 3
10	*	*	10	2 2 1 3 1 2
11	+	+	11	2 3 1 2 1 2
12	,	,	12	1 1 2 2 3 2
13	-	-	13	1 2 2 1 3 2
14	.	.	14	1 2 2 2 3 1
15	/	/	15	1 1 3 2 2 2
16	0	0	16	1 2 3 1 2 2
17	1	1	17	1 2 3 2 2 1
18	2	2	18	2 2 3 2 1 1
19	3	3	19	2 2 1 1 3 2
20	4	4	20	2 2 1 2 3 1
21	5	5	21	2 1 3 2 1 2
22	6	6	22	2 2 3 1 1 2
23	7	7	23	3 1 2 1 3 1
24	8	8	24	3 1 1 2 2 2
25	9	9	25	3 2 1 1 2 2
26	:	:	26	3 2 1 2 2 1
27	;	;	27	3 1 2 2 1 2
28	<	<	28	3 2 2 1 1 2
29	=	=	29	3 2 2 2 1 1
30	>	>	30	2 1 2 1 2 3
31	?	?	31	2 1 2 3 2 1
32	@	@	32	2 3 2 1 2 1
33	A	A	33	1 1 1 3 2 3
34	B	B	34	1 3 1 1 2 3
35	C	C	35	1 3 1 3 2 1
36	D	D	36	1 1 2 3 1 3
37	E	E	37	1 3 2 1 1 3
38	F	F	38	1 3 2 3 1 1
39	G	G	39	2 1 1 3 1 3
40	H	H	40	2 3 1 1 1 3
41	I	I	41	2 3 1 3 1 1
42	J	J	42	1 1 2 1 3 3
43	K	K	43	1 1 2 3 3 1
44	L	L	44	1 3 2 1 3 1
45	M	M	45	1 1 3 1 2 3
46	N	N	46	1 1 3 3 2 1
47	O	O	47	1 3 3 1 2 1
48	P	P	48	3 1 3 1 2 1
49	Q	Q	49	2 1 1 3 3 1
50	R	R	50	2 3 1 1 3 1
51	S	S	51	2 1 3 1 1 3
52	T	T	52	2 1 3 3 1 1
53	U	U	53	2 1 3 1 3 1
54	V	V	54	3 1 1 1 2 3
55	W	W	55	3 1 1 3 2 1
56	X	X	56	3 3 1 1 2 1
57	Y	Y	57	3 1 2 1 1 3
58	Z	Z	58	3 1 2 3 1 1
59	[	[	59	3 3 2 1 1 1
60	\	\	60	3 1 4 1 1 1
61	]	]	61	2 2 1 4 1 1
62	^	^	62	4 3 1 1 1 1
63			63	1 1 1 2 2 4
64	NUL	`	64	1 1 1 4 2 2
65	SOH	a	65	1 2 1 1 2 4
66	STX	b	66	1 2 1 4 2 1
67	ETX	c	67	1 4 1 1 2 2
68	EOT	d	68	1 4 1 2 2 1
69	ENQ	e	69	1 1 2 2 1 4
70	ACK	f	70	1 1 2 4 1 2
71	BEL	g	71	1 2 2 1 1 4
72	BS	h	72	1 2 2 4 1 1
73	HT	i	73	1 4 2 1 1 2
74	LF	j	74	1 4 2 2 1 1
75	VT	k	75	2 4 1 2 1 1
76	FF	I	76	2 2 1 1 1 4
77	CR	m	77	4 1 3 1 1 1
78	SO	n	78	2 4 1 1 1 2
79	SI	o	79	1 3 4 1 1 1
80	DLE	p	80	1 1 1 2 4 2
81	DC1	q	81	1 2 1 1 4 2
82	DC2	r	82	1 2 1 2 4 1
83	DC3	s	83	1 1 4 2 1 2
84	DC4	t	84	1 2 4 1 1 2
85	NAK	u	85	1 2 4 2 1 1
86	SYN	v	86	4 1 1 2 1 2
87	ETB	w	87	4 2 1 1 1 2
88	CAN	x	88	4 2 1 2 1 1
89	EM	y	89	2 1 2 1 4 1
90	SUB	z	90	2 1 4 1 2 1
91	ESC	{	91	4 1 2 1 2 1
92	FS	|	92	1 1 1 1 4 3
93	GS	}	93	1 1 1 3 4 1
94	RS	~	94	1 3 1 1 4 1
95	US	DEL	95	1 1 4 1 1 3
96	FNC 3	FNC 3	96	1 1 4 3 1 1
97	FNC 2	FNC 2	97	4 1 1 1 1 3
98	SHIFT	SHIFT	98	4 1 1 3 1 1
99	CODE C	CODE C	99	1 1 3 1 4 1
100	CODE B	FNC 4	CODE B	1 1 4 1 3 1
101	FNC 4	CODE A	CODE A	3 1 1 1 4 1
102	FNC 1	FNC 1	FNC 1	4 1 1 1 3 1
103	Start A	Start A	Start A	2 1 1 4 1 2
104	Start B	Start B	Start B	2 1 1 2 1 4
105	Start C	Start C	Start C	2 1 1 2 3 2
106	Stop	Stop	Stop	2 3 3 1 1 1 2

Input

- the width in pixels of narrowest bar,
- text to be encoded.

Output

- BMP file containing the barcode image:
  - Sub format: 24 bits RGB – no compression,
  - Image size: 600x50 px,
  - Colors: bars – black, background – white.
- file name: “output.bmp”

Project versions:

1. Code Set A encoding
2. Code Set B encoding
3. Code Set C encoding
4. Encoding of any set (A, B or C).

Remarks:

1. Do not store bars and spaces patterns in coding table as character strings.

References:

[1] BMP file format – see section 4.2
[2] “**Code 128**”, https://en.wikipedia.org/wiki/Code_128
[3] Example images, http://galera.ii.pw.edu.pl/~zsz/ecoar/images/barcodes
[4] Example program for bmp reading/writing,
http://galera.ii.pw.edu.pl/~zsz/ecoar/bmp/bmp_riscv.zip

4 Supplementary materials

This section contains some supplementary materials, which might be useful for your projects.

4.1 Reading data from file in Rars

The code below shows how to read data from a file:

```
#open the file
      li a7, 1024       #system call for file_open
      la a0, fname      #address of filename string
      li a1, 0          # flags: 0-read file
      ecall             #file descriptor of opened file in a0

#save the file descriptor
      mv s1, a0

#check if file was opened correctly (file_descriptor<>-1)
      ...

#read data from file
      li a7, 63         #system call for file_read
      mv a0, s1         #move file descr from s1 to a0
      la a1, buf        #address of data buffer
      li a2, 4048       #amount to read (bytes)
      ecall

#check how much data was actually read
      beq zero,a0, fclose     #branch if no data is read
      ...

#close the file
fclose:
      li a7, 57               #system call for file_close
      mv a0, s1               #move file descr from s1 to a0
      ecall
```

4.2 BMP file header

The contents the header of a 24 bits RGB BMP file is shown in fig. 4.1. A hexadecimal file editor (see remark 4 below) may be used to analyze the contents of binary files. The width and height of the image is stored in 4 byte fields at 0x12 and 0x16 offsets. The offset (address) of the first pixel is stored in 32 bits *Offset of the pixel data* field at 0x0A offset. In example below pixel data starts at 0x36.

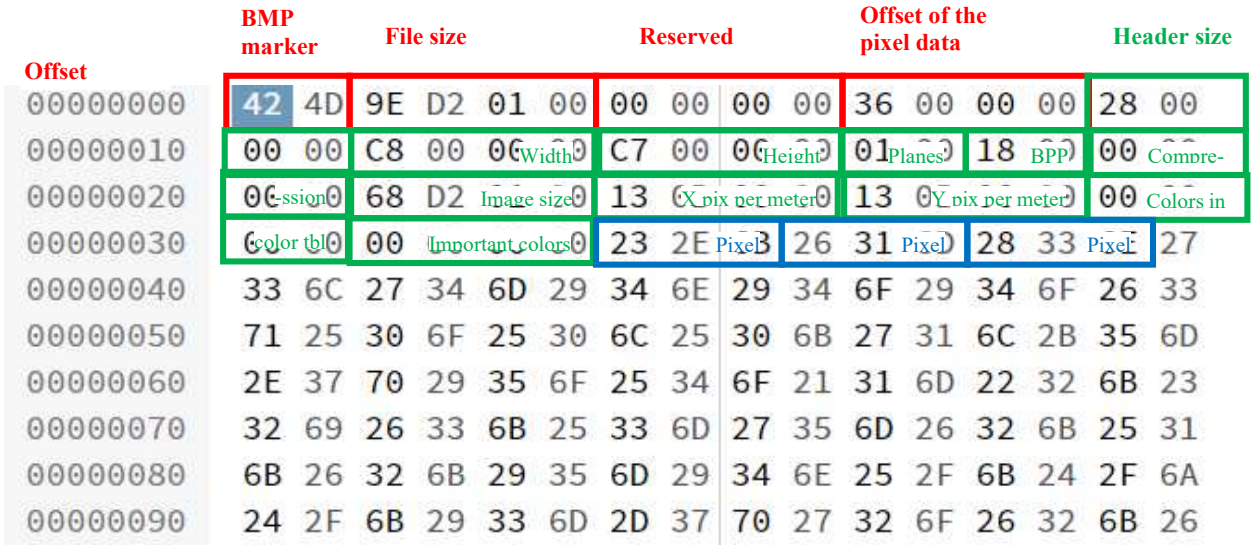

Figure 4.1. Contents of an example BMP file (all numbers in hexadecimal notation).

Remarks:

1. Little endian byte order – the first byte in a field is the least significant one.
2. Order of color components of a pixel: first byte - blue, second byte - green, third byte -red.
3. *Offset of the pixel data* field contains the offset(adress) of the area where the colors of the pixels are stored.
4. Hexadecimal file editor is available at **`https://hexed.it/`**

References:

1. "**file-format-bmp**", https://en.wikipedia.org/wiki/BMP_file_format

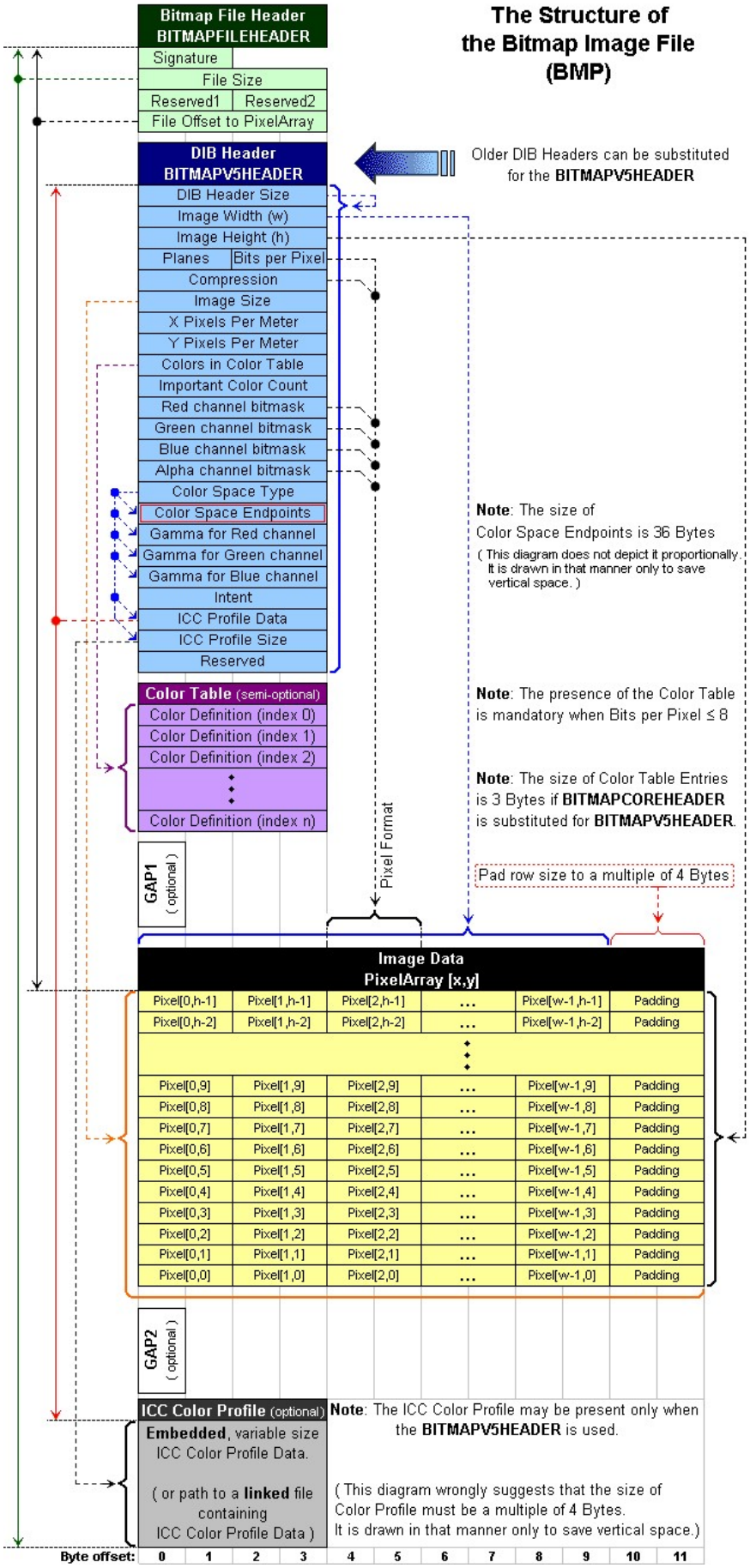

Fig. 4.2. BMP file structure [https://en.wikipedia.org/wiki/BMP_file_format]