Chapter 1

Specification

SPAL simple compile to OpenCL.

1.1 stages of compilation

1. natural language text (perhaps)
2. analytic language text
3. SPAL language text
4. SPAL encoded tiles
5. (OpenCL) C with SPAL names
6. (OpenCL) C with natural names (perhaps)

1.2 Method for implementation

In theory can use any language for implementation. Though ideally would be a version of C which is similar to the above, so it could then be recoded in SPAL.

Pyash	SPAL	C	file
kratta krathnimna li	cardinal _top cardinal name _nom _rea	int main () {	cardinal_name.c
swicta hnimna li	social _top name _nom _rea	void name () {	cardinal_name.c
hmasta hnimna li	mind _top name _nom _rea	inline void name (); inline void name {	library_cardinal_name.h library_cardinal_name.c
krathmasta hnimna li	cardinal mind _top name _nom _rea	kernel void name () {	cardinal_name.cl

htipdoyu txikka hciccu	ten _num _ins indexFinger _acc down _con	`if (i < 0xA) {`
zrundofi	0 _num _return	return 0;
fe	_finally	}
hnimna tyindo cyah	name _nom three _num _cop	`name = 3;`
txikna zrondo cyah	indexFinger _nom zero _num _cop	`i = 0;`
htipdoyu txikka hciccu hyikdoyu plosliwa htekhromli	ten _num _ins indexFinger _acc down _con indexFinger _acc one _num _ins plus _rea _and library program _rea	`for {;i < 0xA; ++i}{` `library_program ();}`

Chapter 2

Pyash Encoding

The virtual machine uses variable-length-instruction-word (VLIW), loosely inspired by head and tails instruction format (HTF). HTF uses VLIW's which are 128 or 256 bits long, however there can be multiple instructions per major instruction word.

2.1 VLIW's Head Index

The head is really a parse index, to show the phrase boundaries. In TroshLyash each bit represents a word, each of which is 16bits, when a phrase boundary is met then the bits flip from 1 to 0 or vice-versa at the phrase boundary word. index takes up the first 16bits of the VLIW. This would lead to 256bit (32Byte) VLIW's. The real advantage of the indexing occurs when there either multiple sentences per VLIW, or when there are complex sentences in the VLIW's. Having the VLIW's broken up into 32Byte chunks, makes it easier to address function entry points, which can be placed at the beginning of a VLIW. Can fit 16 VLIWS in a POSIX page, 128 VLIW's in a Linux page, so would only need 1 byte (8bits) for addressing functions that are within 1 page distance.

2.2 Word Compression

Now for the slightly more interesting issue of packing as many as 5 glyphs into a mere 16 bits. Why this is particularly interesting is that there is an alphabet of 32 glyphs, which would typically required 5 bits each, and thus 25bits in total. However the 16 bit compression is mostly possible due to the rather strict phonotactics of TroshLyash, as only certain classes of letters can occur in any exact place. The encoding supports 4 kinds of words, 2 grammar word classes and 2 root word classes. Where C is a consonant, T is a tone and V is a vowel, they are CVT, CCVT, and CVTC, CCVTC respectively.

2.2.1 CCVTC or CSVTF

I'll start with explaining the simplest case of the CCVTC word pattern. To make it easier to understand the word classes can call is the CSVTF pattern, where S stands for Second consonant, and F stands for Final Consonant. The first C represents 22 consonants, so there needs to be at least 5 bits to represent them. Here are the various classes

"C" :"p","t","k","f", "s","c","x", "b","d","g","v", "z","j", "n","m","q","r", "l","y","w",

"S" "f","s","c","y", "r","w","l","x", "z","j","v",

"V" "i","a","e","o","u","6",

"T" "7","-",

"F" "p","t","k","f", "s","c","n","m"

, (can check the phonology page for pronunciation) C needs 5 bits, S would need 4 bits, however the phonotactics means that if the initial C is voiced, then the S must be voiced, thus "c" would turn into "j", "s" into "z" and "f" into "v", also none of the ambigiously voiced phonemes (l, m, n, q, y, w, r) can come before a fricative because they have a higher sonority, thus must be closer to the vowel. So S only needs 3 bits. V needs 3 bits T needs 2 bits and F needs 3 bits which is a total of 16 bits. 5+3+3+2+3 = 16 However there are other kinds of words also. we'll see how those work.

2.2.2 HCVTF

So here we have to realize that CVC or CVTC is actually HCVTF due to alignment. So what we do is make a three bit trigger from the first word, the trigger is 0, which can be three binary 0's, 0b000 3+5+3+2+3 = 16 H+C+V+T+C this does mean that now 0b1000, 0b10000 and 0b11000 is no longer useable consonant representation, however since there are only 22 consonants, and only 2 of those are purely for syntax so aren't necessary, so that's okay, simply can skip the assignment of 8, 16 and 24.

2.2.3 CSVT

This is similar to the above, except we use 0b111 as the trigger, meaning have to also skip assignment of 15, 23 and 31. $3 + 5 + 3 + 3 + 2 = 16$?+C+S+V+T

2.2.4 CVT

For this one can actually simply have a special number, such as 30, which indicates that the word represents a 2 letter word. 5+5+3+2+1 $F+C+V+T+P$ what is PF P can be a parity-bit for the phrase, or simply unassigned.

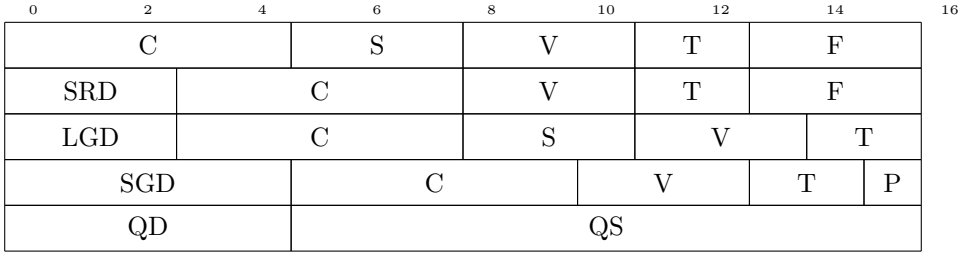
2.3 Quotes

Now with VM encodings, it is also necessary to make reference to binary numbers and things like that. The nice thing with this encoding is that we can represent several different things. Currently with the above words, we have 1 number undefined in the initial 5 bits. 29 can be an initial dot or the final one, can call the the quote-denote (QD), depending on if parser works forwards or backwards. Though for consistency it is best that it is kept as a suffix (final one), as most other things are suffixes. 5+3+8 = 16 Q+L+B QD has a 3 bit argument of Length. The Length is the number of 16bit fields which are quoted, if the length is 0, then the B is used as a raw byte of binary. Otherwise the B represents the encoding of the quoted bytes, mostly so that it is easier to display when debugging. The type information is external to the quotes themselves, being expressed via the available TroshLyash words. So in theory it would be possible to have a number that is encoded in UTF-8, or a string that is encoded as a floating-point-number. Though if the VM interpreter is smart then it will make sure the encoding is compatible with the type Lyash type, and throw an error otherwise.

2.4 Extension

This encoding already can represent over 17,000 words, which if they were all assigned would take 15bits, so it is a fairly efficient encoding. However the amount of words can be extended by increasing number of vowels, as well as tones. And it may even be possible to add an initial consonant if only one or two of the quote types is necessary. However this extension isn't likely to be necessary anytime in the near future, because adult vocabulary goes up to around 17,000 words, which includes a large number of synonyms. For instance the Lyash core words were generated by combining several different word-lists, which were all meant to be orthogonal, yet it turns out about half were internationally synonyms, so were cut down from around eight thousand to around four thousand words. It will be possible to flesh out the vocabulary with compound words and more technical words later on. Also it might make sense to supplant or remove some words like proper-names of countries.

2.5 Encoding Tidbit Overview



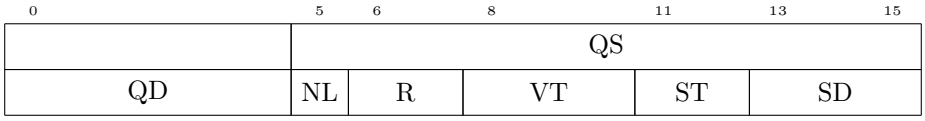
Legend C	Initial Consonant
S	Secondary Consonant
V	Vowel
T	Tone
F	Final Consonant
SRD	Short Root Denote
LGD	Long Grammar Denote
SGD	Short Grammar Denote
P	(optional) Phrase Parity Check tidbit
QD	Quote Denote
QS	Quote Sort 2.7

2.6 Table of Values

#	C	S	V	T	F
width	5	3	3	2	3
0	SRD	y	i	MT	m
1	m	w	a	7	k
2	k	s z	u	_	p
3	y	l	e	U	n
4	p	f v	o		s
5	w	c j	6		t
6	n	r	U		f
7	LGD	x	U		c
8	SRO				
9	s				
10	t				
11	l				
12	f				
13	c				
14	r				
15	LGO				
16	SRO				
17	b				
18	g				
19	d				
20	z				
21	j				
22	v				
23	LGO				
24	SRO				
25	q				
26	x				
27	1				
28	8				
29	QD				
30	SGD				
31	LGO				

	blank means out of bounds
U	undefined
MT	middle tone, no marking
QD	quote denote
SGD	short grammar word denote
SRD	short root word denote
LGD	long grammar word denote
SRO	short root word denote overflow
LGO	long grammar word denote overflow

2.7 Quote Sort



2.7.1 definitions

QS quote sort

QD quote denote

NL name or literal bit

R region

VT vector thick

ST scalar thick

SD sort denote

16 tidbit					
5 tidbit	1 tidbit	2 tidbit	3 tidbit	2 tidbit	3 tidbit
quote denote	literal or name	region	vector thick	scalar thick	sort denote
definitions					
0	literal	private	1	1 byte, 8 tidbit	letter (s)
1	name	world	2	2 byte, 16 tidbit	word (s)
2		eternal	4	4 byte, 32 tidbit	sentence (s)
3		coworker	8	8 byte, 64 tidbit	database
4			16		unsigned integer
5			U		signed integer
6			U		floating point number
7			3		function

The quote denote is 5 bits long, leaving 11 bits. the next 2 bits is used to indicate bit thickness of quote scalar (s), the following 3 bits is used to indicate the magnitude of the vector (s), 1 bit for name or literal

In the case of a refferential, or variable name, the name can be (up to) four words long, that way it fits in a 64bit area — similar to a 64bit address.

2.8 Independent-Clause Code Name

Each independent-clause can have a code name to help find it's program.

There is a mix of grammatical-cases, sorts and a verb in each independent clause. For matching with modern computer processing, a 64bit thickness is desired, though a 128 bit thickness may be possible.

The grammatical cases can have a table to make it easy to identify them.

five bits to designate the case, and 11 bits for the quote type.

The context will henceforward be referred to as scene, and the other half being the posture.

This does mean that any independentClause would have a maximum of three grammatical-cases plus a verb if in 64 bit.

	0 source-case	1 way-case	2 destination-case	3 location-case
0 base	nominative-case	instrumental-case	dative-case	accusative-case
1 space-context (x)	ablative-case	prosecutive-case	allative-case	locative-case
2 genitive-case	possessive-case	descriptive-case	possessed-case	relational-case
3 discourse-context	initiative-case	topic-case	terminative-case	vocative-case
4 social-context	causal-case	evidential-case	benefactive-case	comitative-case
5 surface-context (y)	delative-case	vialis-case	superlative-case	superessive-case
6 interior-context (z)	elative-case	perlative-case	illative-case	inessive-case
7 time-context (t)	initial-time	during-time	final-time	temporal-case

Table 2.1: grammtical-case number system

0	11	14 15
11	3	2
QS	S	P

Table 2.2: grammtical-case code

QS quote sort 2.7

S scene

P posture

0	2	4	6
64 tidbit			
G	G	G	V

0	2	4	6
VUL2 vector of two 64 tidbit numbers for searching			
or VUS8 vector of eight 16 tidbit numbers for establishing			
G	G	G	G
V	V xor T	V xor A	Pe

Table 2.3: code name sketch

G grammatical-case

V verb

T tense

A aspect

Pe perspective (grammatical-mood)

Can also have seperate identifiers for the verb and the grammatical-cases, then it would be easier to have multi-word verbs.