Henry is in the common room: on the coffee table in front of him are some large sheets of paper with handwriting on them - Henry's handwriting - and a cup of peppermint tea.

Henry is engrossed and looking at the papers and thinking.

Edith enters.

"Hello Henry, you look busy. Something interesting?"

"Hello Edith: yes, it is just something that I am thinking about in my tea break. I suppose that it could be regarded as a sort of mathematical toy."

"Do you regard it as a mathematical toy?" asks Edith, smiling as she asks.

Henry pauses.

"You read me like a book Edith. Actually I am hoping that it could be very useful in practice, yet it is something of a leap forward."

"So you say it could be regarded as a sort of mathematical toy so as perhaps to forestall criticism - because if it is only a toy - well ...."

"As I say Edith, you read me like a book."

"So would you be willing to explain the idea to me, just as a chat over a cup of peppermint tea?"

"Well, um .... alright."

"I'll just go and make a cup of peppermint tea and then I'll come back and listen while the tea cools before I drink it - I promise that I won't criticise."

"Right." says Henry, as he starts to think of how to explain the idea to Edith.

Edith has returned from the peninsula unit in the corner of the room with a cup of peppermint tea.

"Well, the idea is to have a portable interpretable object code for a virtual computer. Yet also that that portable interpretable object code can be encoded unambiguously into plain text."

"So how would that encoding work - in general terms?" asks Edith.

"Well, a piece of object code could be encoded by using a base character followed by a sequence of tag characters. A similar sort of encoding to that which we use for localizable sentences. The difference being that a different base character is used."

"Ah. What base character are you using?"

"Well, at the moment I am using a Private Use Area character, F47F0. So as it is a Private Use Area character the assignment is not unique, but, for the moment, it will do."

"While it is just a mathematical toy?" smiles Edith.

Henry looks at Edith and smiles, with some embarrassment.

“It’s alright, I’m not going to criticise. You are using a Private Use Area characters in order to be able to try out the idea, that is fine. Is there any particular reason for that particular character?”

“Well, yes actually. The F47F0 is hexadecimal: the base ten equivalent is 1001456 so that makes it easy to remember, one million and fourteen fifty-six, because the system is based on an idea from some years ago named fourteen fifty-six object code. Using the base ten number makes it easier to enter the character into the computer using an Alt code.”

“Ah. .... You referred to a virtual computer: what is that please?”

“It is a software construct. Any particular use of the virtual computer on any particular type of physical computer would require software written to run on that particular type of physical computer so as to simulate in software the operations of the virtual computer, so that the software for any particular program that runs on the virtual computer is always the same and always produces the same result regardless of what is the underlying type of physical computer.”

“Is the software like the machine code for a microprocessor?” asks Edith.

“Well, yes in some ways, though not in others. It is an object code. This object code is strongly typed. There are commands for integers, for double precision floating point numbers, for characters, for strings, for complex numbers, for quaternions. I am thinking about whether there could be some types specifically for processing items found in OpenType fonts, such as a type each for point, contour and glyph. Then a point would be x and y and whether the point is on-curve or off-curve, a contour would be a sequence of points and a glyph would be a collection of one or more contours. Then it would be possible to read in several glyphs, scale them and move them horizontally and vertically and then combine them all to produce a new glyph.”

“That sounds interesting.”

“Thank you.”

“I appreciate that you are still looking at the idea, and that it could remain as just some sort of mathematical toy, but do you have any ideas as to whether it would be useful in practice?”

“Well, it could be useful for a very advanced OpenType rendering in that maybe an OpenType rendering system could be programmable, from a font, from within a document where some special project is being discussed or from a stand-alone file of plain text. Yet that is for the future.”

“So, can you give me an example of a command for the virtual computer and how it should be interpreted please?”

“Yes, certainly. Each command for the virtual computer would be either one character or two characters. That is so that there can be more possible commands than there could be if every command were expressed using just one character. As a specific example,

there would be the command I+ which is a two character command. The I+ is for adding two integers and it also sets the value of the second integer to zero.

Two registers are used, the integer accumulator ai and an integer register bi.

The effect can be expressed as follows.

{ai:=ai+bi; bi:=0;}

The bi register is cleared as it is mostly used just as a place to store an integer value before it is used. The content could have been entered just as a literal value directly from the software, or it could have been recovered from the integer memory. The value in bi could have been gathered from a variety of places and it could be used in a variety of ways, for example with an I- command or an I* command and so on."

I+ {ai:=ai+bi; bi:=0;}

I- {ai:=ai-bi; bi:=0;}

I* {ai:=ai*bi; bi:=0;}

I/ {ai:=ai div bi; bi:=0;}

I% {ai:=ai mod bi; bi:=0;}

"Yet for the avoidance of doubt I emphasise that ai and bi are not the accumulator of the physical computer that would be running a program that implements the virtual computer."

Edith nods.

"You mentioned having Glyph as a type. How would that work please?" asks Edith.

"Well, the commands would be similar to those for Integer commands, though there would be a different set of commands. For example, as I have the design at the moment there would be a G+ command for adding two glyphs together."

G+ {ag:=ag+bg; bg:=empty;}

"That is very similar in form to the I+ command except that it operates on glyphs. The idea is that the software in the interpreting program would do the quite large computation behind the scenes, so that the person writing a particular program to run on the virtual computer would just use G+ to add the two glyphs together. There would not be G- G* G/ and G% commands as the meanings of such commands is not determined. There would however be a Gs command to scale a glyph and there would be a Gh command to move a glyph horizontally and there would be a Gv command to move a glyph vertically. So if one wanted to gather two glyphs from a font and for each of them scale the glyph down in size and then move the glyph, and then add the two glyphs together to produce a new glyph, then that would be possible. Or more than two glyphs if one so chooses."

"So, for moving a glyph horizontally, how would that work?"

“Well the command would be Gh and that would move all of the points in the glyph that is in the ag register to the right by a number of font units. The specific number being taken from the ai register, and that number could be positive or negative, a negative number moving the glyph to the left.”

“That is fascinating.”

“Thank you.”

“Do you have a list of the various types that you are considering using at the moment?”

“Yes, here is a list.”

I Integer

D Double precision floating point

B Boolean

H Character

S String

Z Complex

Q Quaternion

P Point, as in a font, a point is a triple of two Integers and a Boolean

C Contour, a sequence of P items

G Glyph, a sequence of C items

and L is used for commands for the Link flag.

“The Link flag? What is that please?”

“Ah, it is part of the structure of the 1456 object code. There are two jump commands in 1456 object code, there is J which is unconditional and !J which is conditional and only takes place if the Link flag is set as true. There are commands, for some types, that compare the contents of the a register and the b register and then set the Link flag depending on the result.”

“The document that I am working from has the following.

quote

The linkflag1456 is now introduced, which is a boolean value which is either true or false. The Jump on linkflag1456 being true command is !J and the Call subroutine on linkflag1456 being true command is !C and the Return on linkflag1456 being true is !R.

end quote

However, for this system which I am designing now, which is based on the 1456 object code from the year 2000, I am trying to keep capital letters for types as far as possible. Also I want to use C for a contour type. So I am thinking of having the following."

Henry points to a handwritten note that he has made.

The Jump command is j, the Call command is c and the Return from subroutine command is r. The Jump on linkflag being true command is !j and the Call subroutine on linkflag being true command is !c and the Return on linkflag being true is !r.

Edith reads the note.

"So what commands can set the linkflag one way or the other?" asks Edith.

"Well, there are quite a lot of them. For example, I> will set the linkflag to true if the contents of ai is greater than the contents of bi and I= will set the linkflag to true if the contents of ai is equal to the contents of bi, and I< will set the linkflag to true if the contents of ai is less than the contents of bi."

"There are also these."

Lt linkflag:-true;

Lf linkflag:=false;

Ln linkflag=not linkflag;

"So, for example, testing for less than or equal could be done by two commands in sequence, namely I> followed by Ln as less than or equal to is the same as not greater than."

"Yes."

"Also there would need to be a command to mean get a glyph from the font using a code number and place it into the bg register: also a command to mean to use the glyph in ag as the output glyph. The virtual computer for building into the OpenType rendering system would probably not include quaternions and might not include complex numbers, though they could perhaps be useful for rotating glyphs if that facility is required."

"This is fascinating." opines Edith.

"I am hoping that it will be of practical use eventually. At the moment it is just something that is interesting to think about at times." comments Henry.