- The goal of project a03 is to produce a P-Code executable program that can be executed by the P-Machine.
- A P-Code executable program is formatted according to the specifications given by the P-Machine Executable Format (http://cs2.uco.edu/~trt/cs4173/pexecutable.pdf)
- A P-Machine Executable Program file is **NOT** a text file: it is a binary file. The executable program cannot be read by *cin* nor printed by *cout*.
- A P-Machine Executable Program has three (3) parts
    1. A directory providing access to the remaining components
    2. Constants, including integer, real, set, and string constants
    3. P-Code instruction array that contains the actual program

Directory:

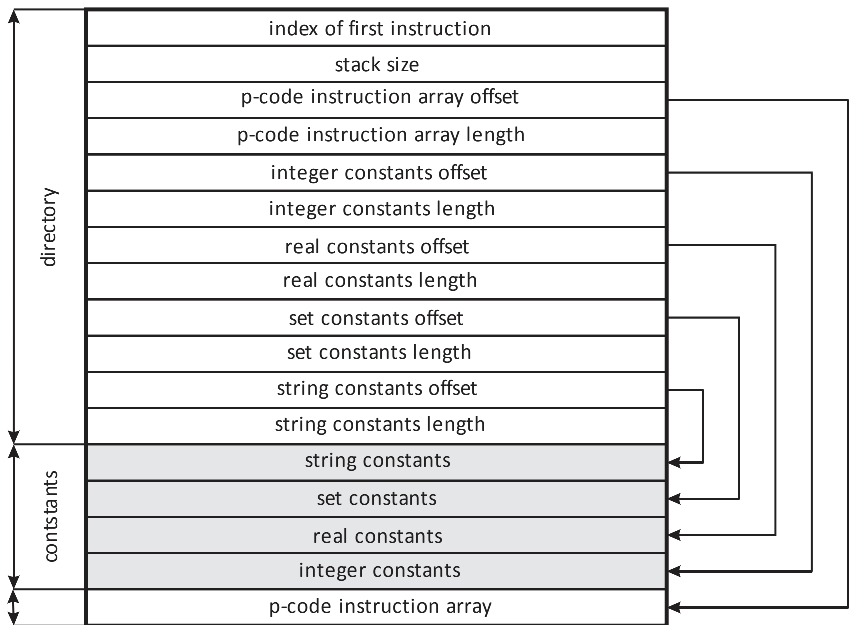

Figure 1. Anatomy of a P-Machine Executable Program File

String Constants:

String constants are stored in an array. String constants vary in size. P-Machine strings have the same form as C-programming language strings. Strings consist of a sequence of characters terminated by a character whose ordinal value is zero. P-Machine instructions reference string constants by using the index of the first character of the string.

Two strings are shown in Figure 2. The string "candy" is followed by the string "is." Together, both strings require nine (9) characters including their terminators.

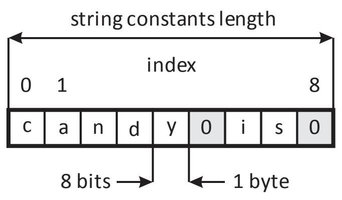

Figure 2. Anatomy of string constants

Set Constants:

Set constants are stored in an array. Each set occupies 64 bits. The number of sets, n_t, is given by n_t=l_t/8, where l_t is the value of the field labeled "set constants length" in the directory.

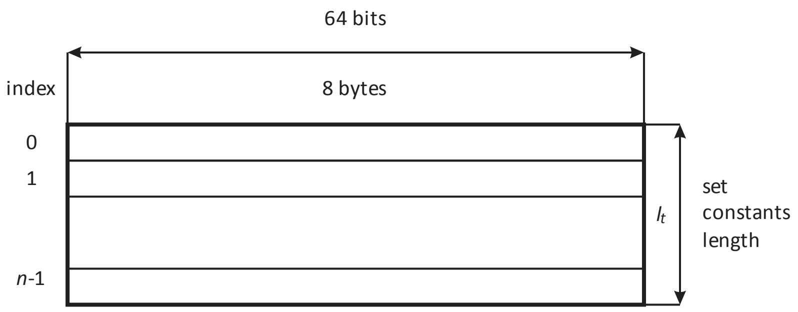

Figure 3. Anatomy of set constants

1. Real constants

Real constants are stored in an array. Each real number occupies 64 bits. The number of real constants, n_r, is given by, n_r=l_r/8, where l_r is the value of the field labeled "real constants length" in directory. Real constants are stored in IEEE 754 binary double format.

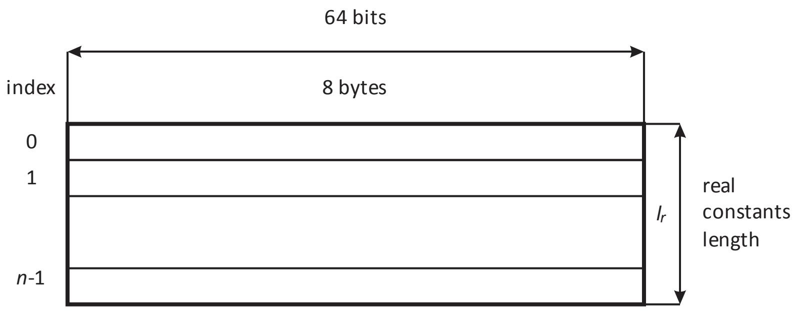

Figure 4. Anatomy of real constants

2. Integer constants

Integer constants are stored in an array. A standard integer occupies four bytes (32 bits) on Intel Computers used by the Department of Computer Science at the University of Central Oklahoma. The number of integers, n_i is given by $n_i = l_i/4$ where l_i is the value stored in the directory in the field labeled "integer constants length."

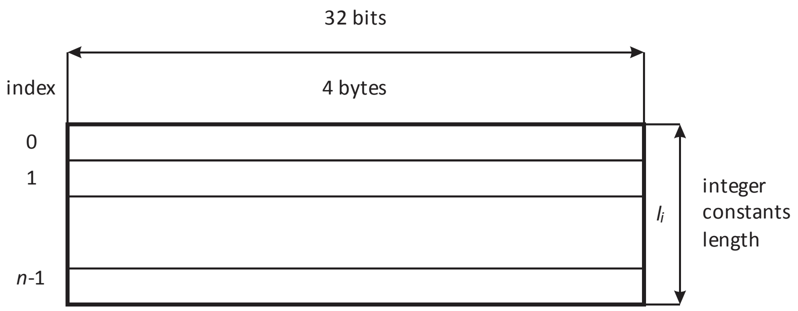

Figure 5. Anatomy of integer constants

P-Code instruction array

P-Code instructions are stored in an array. Each P-Code instruction occupies 4 bytes (32 bits). The number of P-Code instructions, n_p, is given by, $n_p = l_p/4$, where l_p is the value of the field labeled "P-Code instructions array length" in the directory.

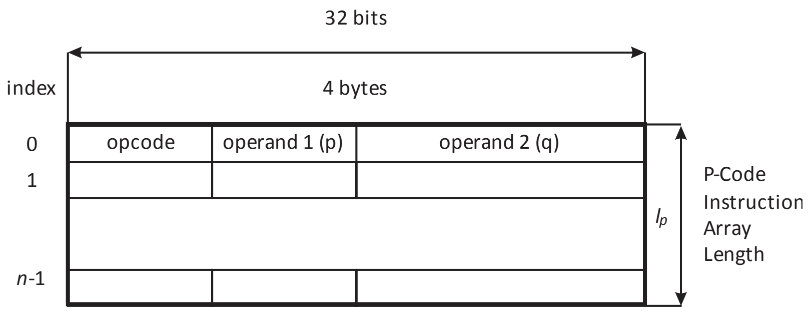

Figure 6. Anatomy of the P-Code instruction array

P-Code instructions consist of three fields, an operation code (opcode) followed by two operands. Operands depend on the opcode.

P-Code instructions occupy 32 bits or 4 bytes. The opcode is stored in the first byte of the instruction. The first operand, called *p*, is stored in the second byte and the second operand, called *q*, occupies the remaining two bytes as shown in Figure 7.

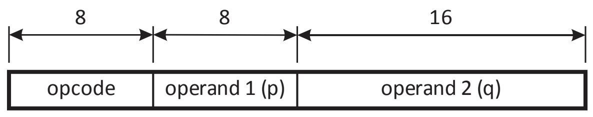

Figure 7. Anatomy of a P-Code instruction

The opcode and both operands are unsigned integers.

Writing the P-Machine Executable Program File Directory:

```
//--------------------------------------------------------------------------
//Function Write writes the internal (unformatted) representation of the
//directory to file f.
//--------------------------------------------------------------------------
    void PasmDirectory::Write(FILE* f)
    {  fwrite(this,sizeof(PasmDirectory),1,f);
    }
```

```
class PasmDirectory {
  int start;             //Index of the first instruction
  int ssize;             //Stack size
  int iaoffset;          //Instruction array offset
  int iasize;            //Instruction array size
  int icoffset;          //Integer constants offset
  int icsize;            //Integer constants size
  int rcoffset;          //Real constants offset
  int rcsize;            //Real constants size
  int scoffset;          //String constants offset
  int scsize;            //String constants size
  int tcoffset;          //Set constants offset
  int tcsize;            //Set constants size
public:
  PasmDirectory();
  ~PasmDirectory();
  void StoreSizes             //Stores sizes of regions specified in the
                              //directory
    (int sts                  //Stack size
    ,int ias                  //P-Code instruction array size
    ,int ics                  //Integer constants size
    ,int rcs                  //Real constants size
    ,int scs                  //String constants size
    ,int tcs                  //Set constants size
    );
  void Print(ostream& o);
  void Write(FILE* f);
  void Read(FILE* f);
};
```

Writing the P-Machine Executable Program Integer Constants:

```
//------------------------------------------------------------------------
//Function Write writes the binary image of theinstruction array to file f
//------------------------------------------------------------------------
void PasmIntegerConstants::Write(FILE* f)
{  fwrite(I,sizeof(int),count,f);
}
```

Writing the P-Machine Executable Program P-Code Instruction Array:

```
//------------------------------------------------------------------------
//Function Write writes the binary image of theinstruction array to file f
//------------------------------------------------------------------------
void PasmInstructionArray::Write(FILE* f)
{  for (int a=0;a<count;a++) IA[a].Write(f);
}
```