Bit pattern diagrams

Jean-Marc Bourguet
jm@bourguet.org

December 6, 2015

Abstract

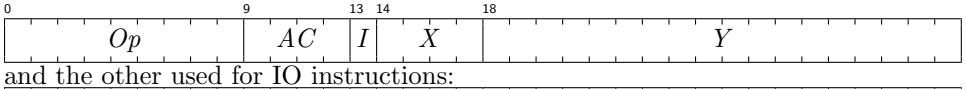
The `bitpattern` package is designed to typeset bit patterns as they may appear in description of data format, hardware or protocols. It covers thus more or less the same application domain as the package `register` and is somewhat related to `bytefield`.

Comparared to `register` the formating is more compact, the syntax less verbose and `bitpattern` allows big endian and little endian bit numbering. But `bitpattern` is less well adapted to the use of long names for the fields and has no provision for a reset value.

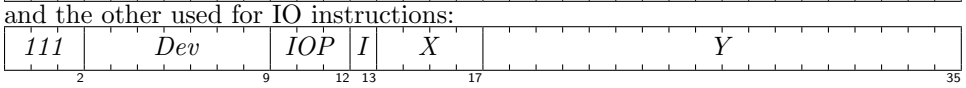
1 Examples

We'll first use the instruction formats of the PDP-10 to describe the features of bit patterns. The PDP-10 was a word-adressable 36-bit computer and the reason for which I've chosen it is that it was the computer on which TEX was first implemented.

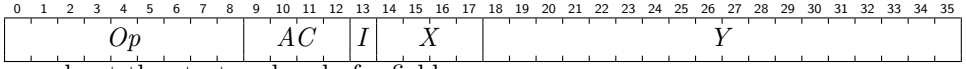
The PDP-10 has two instruction formats, one used for most of the instructions:



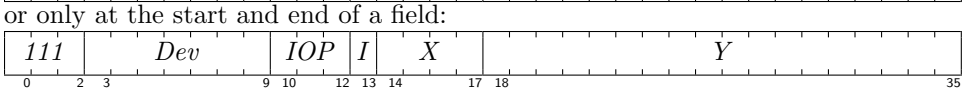
and the other used for IO instructions:



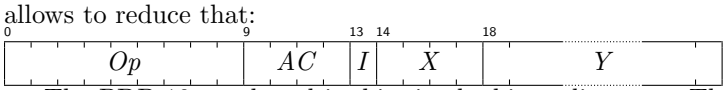
`bitpattern` allows to number all the bits:



or only at the start and end of a field:

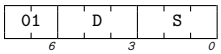


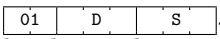
Having a long field like Y here may take more place than wanted. `bitpattern` allows to reduce that:



The PDP-10 numbered its bits in the big endian way. That's not the case of the 8080. The 8080 was an 8-bit computer. Trying to pack instructions in 8-bit forces to use a lot of formats. One of them was used for the move instruction,

and let's take that opportunity to start and compact the layout of the bit field by changing the format of the bit numbers and the field descriptions:



After that it is possible to still be more compact by removing the numbers and reducing the height of the ticks. When that's done, you can put a bit pattern in a paragraph without disturbing it too much, like this 

. With some text afterwards to see how the next line is handled. The result seems quite readable.

2 Interface

2.1 `\bitpattern`

`\bitpattern` The `\bitpattern` macro is the macro which formats the patterns.

`\bitpattern[`⟨*format*⟩`]{`⟨*name*⟩`}[`⟨*size*⟩`][`⟨*width*⟩`]...\`

The optional argument allowing to control how the formatting is made:

littleEndian indicates that bit numbering is little endian, that is the leftward bit has the biggest number;

bigEndian indicates that bit numbering is big endian, that is the leftward bit has the lowest number;

numberBitsAbove indicates that the bit numbers should be put above the fields;

numberBitsBelow indicates that the bit numbers should be put below the fields;

noBitNumbers indicates that there should be no bit numbers;

numberOncePerField indicates that the fields should have only one bit number;

numberTwicePerField indicates that the fields should have two bit numbers;

numberAllBits indicates that the fields shouldn't have a bit number indication;

startBit=X indicates that the number for the leftmost bit should be X;

bitWidth=dimen indicates that the size taken by a bit should be *dimen*;

tickHeight=dimen indicates that the size taken by the small ticks marking the bits should be *dimen*.

After the optional arguments comes the description of the fields. They are composed of the field names (which does not have to be included in braces if it takes one character) followed by two optional arguments giving the size of the field (1 if omitted) and the width it should take (the same as its size).

Field descriptions are ended by a `/`.

2.2 Package options

The package `bitpattern` accept some options:

littleEndian indicates that the default for bit numbering is little endian, that is the leftward bit has the biggest number;

bigEndian indicates that the default for bit numbering is big endian, that is the leftward bit has the lowest number;

numberBitsAbove indicates that by default the bit numbers should be put above the fields;

numberBitsBelow indicates that by default the bit numbers should be put below the fields;

noBitNumbers indicates that by default there should be no bit numbers;

numberOncePerField indicates that by default the fields should have only one bit number;

numberTwicePerField indicates that by default the fields should have two bit numbers;

numberAllBits indicates that by default the fields shouldn't have a bit number indication.

2.3 Commands controlling the format

`\bpLittleEndian` `\bpLittleEndian` changes the default bit numbering to little endian.

`\bpBigEndian` `\bpBigEndian` changes the default bit numbering to little endian.

`\bpNumberBitsAbove` `\bpNumberBitsAbove` changes the default to having the numbering above the fields.

`\bpNumberBitsBelow` `\bpNumberBitsBelow` changes the default to having the numbering below the fields.

`\bpNoBitNumbers` `\bpNoBitNumbers` changes the default to having no numbering.

`\bpNumberFieldOnce` `\bpNumberFieldOnce` changes the default to having the numbering done once per field.

`\bpNumberFieldTwice` `\bpNumberFieldTwice` changes the default to having the numbering done twice per field.

`\bpNumberAllBits` `\bpNumberAllBits` changes the default to having the numbering done for all bits of a field.

`\bpStartAtBit{`⟨*number*⟩`}`

`\bpStartAtBit` `\bpStartAtBit` gives the default bit number of the leftmost bit.

`\bpSetBitWidth{`⟨*length*⟩`}`

`\bpSetBitWidth` `\bpSetBitWidth` gives the default bit width

\bpSetTickHeight{⟨*length*⟩}

\bpSetTickHeight — \bpSetTickHeight gives the default height for the ticks marking the bits in a multi-bit field.

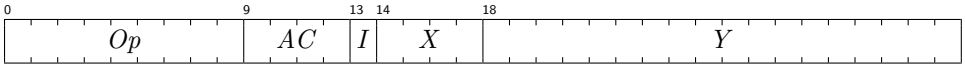
\bpFormatField — \bpFormatField is a one argument macro used fo format the field. It can be replaced. Care should be taken to format all the fields with the same height, so putting a \strut in the replacement is probably in order.

\bpFormatBitNumber — \bpFormatBitNumber is a one argument macro used fo format the bit numbers. It can be replaced. Care should be taken to format all the bit numbers with the same height, so putting a \strut in the replacement is perhaps in order.

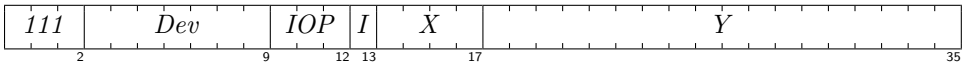
3 Examples revisited

bitpattern was loaded by:

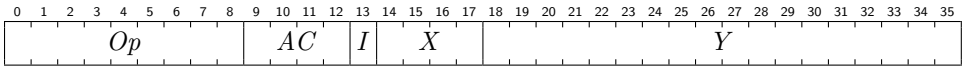
```
\RequirePackage[numberOncePerField,bigEndian,numberBitsAbove]{bitpattern}
```



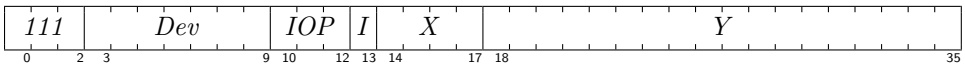

```
\bitpattern{Op}[9]{AC}[4]IX[4]Y[18]/
```



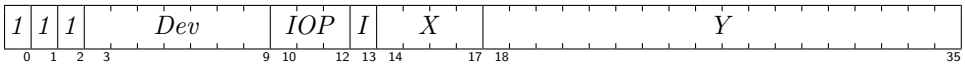

```
\bitpattern[numberBitsBelow]{111}[3]{Dev}[7]{IOP}[3]IX[4]Y[18]/
```



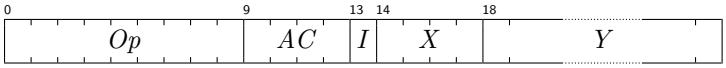

```
\bitpattern[numberAllBits]{Op}[9]{AC}[4]IX[4]Y[18]/
```



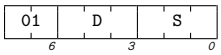

```
\bitpattern[numberBitsBelow,numberTwicePerField]%
      {111}[3]{Dev}[7]{IOP}[3]IX[4]Y[18]/
```



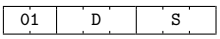

```
\bitpattern[numberBitsBelow,numberTwicePerField]%
      111{Dev}[7]{IOP}[3]IX[4]Y[18]/
```




```
\bitpattern{Op}[9]{AC}[4]IX[4]Y[18][9]/
```




```
\renewcommand\bpFormatBitNumber[1]{{\tiny\ttfamily\emph{\strut#1}}}
\renewcommand\bpFormatField[1]{{\scriptsize\ttfamily\strut#1}}
\bitpattern[littleEndian,startBit=7,numberBitsBelow]{01}[2]{D}[3]{S}[3]/
```




```
\bpSetTickHeight{1pt}
\bitpattern[noBitNumbers,littleEndian,startBit=7]{01}[2]{D}[3]{S}[3]/
```

It is to be noted that usually one does not change the formatting for every pattern, so setting the optional argument of `\bitpattern` is rarely used.