MemWalker

Memwalker is a simple program, which can be used to dump registers memory wlong with its bit meanings.
Memwalker reads the register values using /dev/mem.

Where and why this memwalker is useful:

- It is coded in C language.
- It can be compiled on the embedded platform which is running linux.
- Also, since python compiler is not present always, it is just easy to use this kind of tool.
- You can also use this tool to check bits and its meaning even if you have register dump from some other sources. The app can map the provided values to the register and will provide its bits meanings and division..

The MemWalker needs two files:

a.SOC Register description file:

It contains the register descriptions for the cpu/soc

File structure:

```
SOC_Name
{
        register_name register_address register_bit_size
        {
                BIT_name       BIT_TO    BIT_FROM     defaultval #comment
        }
}
```

Description of bit fields/Comments are marked with a previous '#' and allowed at a start of line, or at
end of a line.

Example: (*soc_ls2088.reg*)

```
LS2088
{
        SEC_MCFGR       0x8180270       32
        JRSTARTR        0x8180288       32
        {
                RSVD            31      16      0       #rsvd
                START_ADDR      15      0       0       #job Ring start
        }
        SEC_STATUS      0x8180290       32
        QMAN_STATS      0x81802A0       32
        {
                ENABLED         31      31      0 # 1- enabled, 0 -disabled
                RSVD            30      30      0 # reserved bytes
                EQ_RJ           29      26      f # Enqueue rejections count
                CGR_CNT         25      20      3f # CGR Count
                FIFO_FULL       19      17      7 # FIFO Full count
                EQ_WRED         16      13      f # Enqueue WRED rejections count
                EQ_SUCCESS      12      7       3f # Enqueue SUccess count
                EQ_TD           6       4       7 # Enqueue Tail drop count
                RSVD2           3       2       3 # Reserved
                STATUS          1       0       0 #  Status
        }
        QMAN_FQ_STATUS  0x81802A8       32
        BMAN_STATS      0x81802B0       32
        {
                B_FREE  31      16      12 # Total free buffers availaible in bpool
                BTOTAL  15      0       13 # total buffers in bpool
        }
}
```

b. Memory to be walked file:

a simple list of registers whose dump needs to be taken and shown with bitwise description.

Structure of file:

registername1
registername1
.
.
registernameX

where:
  registernameX: name of the register defined in the
                 SOC Register description file.

Example: (*walk_these.reg*)

```
QMAN_STATS
BMAN_STATS
```

Note: You can also use the application to get register bit meanings, if you have got some register dump from some other source.

Then, the structure of this file will be like this :

Structure of file:

```
registername1   <value1>
registername1   <value2>
.
.
registernameX   <valueX>
```

where:

```
  registernameX: name of the register defined in the
                 SOC Register description file.

  valueX : value of the register memory which needs to analysed
               according to the SOC register description file.
```

Example: (*walk_these_values.reg*)

```
QMAN_STATS     0x12334098
BMAN_STATS     0x220a311c
```

How to compile:

Run the following command on any platform running linux:

```
gcc -w memwalker.c -o memwalker
```

How to run:

```
./memwalker <soc_register_description file> <Memory to be walked file>
```

Sample Output:

With the above two sample files, Memwalker gives the following output:

```
$ ./memwalker soc_ls2088.reg walk_these.reg

--------------------------------------
QMAN_STATS@0x81802a0 32 bit val: 0x770a691f
ENABLED (31..31) :  0   # 1- enabled, 0 - disabled
RSVD (30..30) :  1   # reserved bytes
EQ_RJ (26..29) :  d   # Enqueue rejections count
CGR_CNT (20..25) :  30   # CGR Count
FIFO_FULL (17..19) :  5   # FIFO Full count
EQ_WRED (13..16) :  3   # Enqueue WRED rejections count
EQ_SUCCESS (7..12) :  12   # Enqueue SUccess count
EQ_TD (4..6) :  1   # Enqueue Tail drop count
RSVD2 (2..3) :  3   # Reserved
STATUS (0..1) :  3   #  Status

--------------------------------------
BMAN_STATS@0x81802b0 32 bit val: 0x770a691f
B_FREE (16..31) :  770a   # Total free buffers available in bpool
BTOTAL (0..15) :  691f   # total buffers in bpool
```

With the above two sample files with register values already available, Memwalker gives the following output:

```
$ ./memwalker soc_ls2088.reg walk_these_values.reg

--------------------------------------
QMAN_STATS@0x81802a0 32 bit val: 0x12334098
ENABLED (31..31) :  0   # 1- enabled, 0 - disabled
RSVD (30..30) :  0   # reserved bytes
EQ_RJ (26..29) :  4   # Enqueue rejections count
CGR_CNT (20..25) :  23   # CGR Count
FIFO_FULL (17..19) :  1   # FIFO Full count
EQ_WRED (13..16) :  a   # Enqueue WRED rejections count
EQ_SUCCESS (7..12) :  1   # Enqueue SUccess count
EQ_TD (4..6) :  1   # Enqueue Tail drop count
RSVD2 (2..3) :  2   # Reserved
STATUS (0..1) :  0   #  Status

--------------------------------------
BMAN_STATS@0x81802b0 32 bit val: 0x220a311c
B_FREE (16..31) :  220a   # Total free buffers available in bpool
BTOTAL (0..15) :  311c   # total buffers in bpool
```