Subject: MCR10151, Add ability to coreload.pl1 to generate a MIF file

Author: Dean S. Anderson

Date: 03/09/2025

Multics Ticket: https://multics-trac.swenson.org/ticket/349

In order to support the FPGA development of the DN6678 series front-end processor, test and diagnostic programs need to be written. Since Multics has an assembler for this platform (MAP355), it works well to assemble these programs and eliminates the need to develop a cross-assembler.

The FPGA development software (Quartus) requires a file in MIF (Memory Image File) format (see https://www.intel.com/content/www/us/en/programmable/quartushelp/17.0/reference/glossary/def_mif.htm for details) to be able to load these programs into simulated ROM.

There is already a program, coreload, in Multics that takes a MAP355 object deck and converts it into a core image. A fairly simple modification to this program allows it to also write a MIF file at the same time.

Proposed Changes

All of the changes below are to coreload.pl1.

Add the history comment:

Inserted in B:

```
B11
B12     /****^  HISTORY COMMENTS:
B13       1) change(2025-03-10,Anderson), approve(2025-03-10,MCR10151):
B14         Added generation of a Memory Image File (MIF) to aid in FPGA development.
B15                                        END HISTORY COMMENTS */
B16
```

Preceding:

```
A11     coreload: proc;


A17       Re-written by Mike Grady 5/21/76 to make better. */
```

Changed by B to:

```
B23       Re-written by Mike Grady 5/21/76 to make better.    */
```

Add some working storage to format the MIF lines:

Inserted in B:

```
B35     dcl  lname char (168);
B36     dcl  octal_loc char(6) varying;
B37     dcl  padded_loc char(5);
```

Preceding:

```
A29     dcl  eofsw bit (1);
```

Define an output stream for the MIF output:

Inserted in B:

```
B68     /* Streams */
B69
B70     dcl  mif_output_stream file stream output;
B71
```

Preceding:

```
A59     /* External Entries */
```

We need to use the numeric_to_ascii_base_ entry point so define it:

Inserted in B:

```
B88     dcl  numeric_to_ascii_base_ entry (float dec(59), fixed bin, fixed bin) returns (char(72) varying);
```

Preceding:

```
A75     \014
```

We have done enough parsing and validation to open the MIF output stream:

Inserted in B:

```
B116            /* Open the stream for the MIF file output */
B117            lname = "vfile_ " || substr (out_name, 1, index (out_name, " ") - 1) || ".mif";
B118            open file(mif_output_stream)
B119                title(lname)
```

```
B120            output;
B121
Preceding:
A102        call gcos_gsr_read_$gsr_read_init ("in", code);   /* init reading */
```

Just before the main loop we write some needed header info to the MIF stream:

Inserted in B:

```
B124
B125        /* Write the initial file header lines for the MIF file */
B126        put file(mif_output_stream) edit("WIDTH=18;")(a);
B127        put file(mif_output_stream) skip edit("DEPTH=32768;")(a);
B128        put file(mif_output_stream) skip edit("ADDRESS_RADIX=OCT;")(a);
B129        put file(mif_output_stream) skip edit("DATA_RADIX=BIN;")(a);
B130        put file(mif_output_stream) skip edit("CONTENT BEGIN")(a);
B131
Preceding:
A104    loop:    call gcos_gsr_read_ ("in", bufp, reclen, rcrdhdr, eofsw, code); /* get a record */
```

In the main loop, we check each word and write a MIF record if the word is not zero

(there is no need to write zero records as memory is auto-filled with zeros in the FPGA load):

Inserted in B:

```
B162            if (words (i) ^= "0"b) then do;        /* we can skip output of zero words */
B163              octal_loc = rtrim (numeric_to_ascii_base_ ((loc), -5, 8), "o");
B164              padded_loc = substr ("00000" || octal_loc, length ("00000" || octal_loc) - 4, 5);
B165                        put file(mif_output_stream) edit ("    ", padded_loc, " : ", words (i), ";")
(skip,a,a,a,b(18),a);
B166            end;
Preceding:
A134            loc = loc + 1;
```

We need to write a MIF trailer and close the MIF output stream.

```
A140     finis:    seg.count = divide (loc + 1, 2, 17, 0);
```

Changed by B to:

```
B173     finis:    put file(mif_output_stream) edit("END;")(skip,a);
B174               close file(mif_output_stream);
B175               seg.count = divide (loc + 1, 2, 17, 0);
```

Comparison finished: 9 differences, 39 lines.

Testing

1. The unmodified program was used to generate a core image of a test program.
2. The modified program was used to generate a core image and MIF file of the same test program.
3. The two core images were compared and no differences were found.
4. The MIF file generated was loaded into the FPGA DN6678 implementation and was successfully run.

Documentation

There are no documentation changes as there doesn't appear to be an info page for coreload.

Version History

Date	Revision	Author	Comment
2025-03-08	0.1	Dean S. Anderson	Initial draft
2025-03-10	0.2	Dean S. Anderson	Fixed header; Changed to add comment with hcom.
2025-03-15	1.0	Dean S. Anderson	Updated name of output stream to mif_output_stream and published.