GNU Tools for e200 processors

Release Notes

Table of Contents

1 Release description

This release of gcc for e200-VLE supports the "Power Architecture® 32-bit Application Binary Interface Supplement 1.0 - Embedded". It is based on gcc 4.9.4, binutils 2.28 and gdb 7.8.2.

The current release supports VLE codegen for

- e200z0
- e200z2
- e200z3
- e200z4
- e200z6
- e200z7

It has built-in MULTILIB support for the e200 with soft-float or the Embedded FPU, libraries also support the SPE/SPEv2. SPE and SPEv2 libraries support only Z3, Z4, Z6 and Z7 cores.

This distribution provides support for newlib and newlib-nano. Freescale EWL2 libraries are distributed independently as a part of S32 Design Studio IDE. Since distinct libraries are provided, the tools require setting explicitly `--sysroot` to perform MULTILIB resolution:

For newlib

```
--sysroot={INSTALL}/powerpc-eabivle/newlib
```

For EWL2

```
--sysroot={INSTALL}/e200_ewl2
```

This distribution provides tools for 32-bit host system. For 64 bit system, 32 bit libraries are required to run the tools. See Chapter 7 [Installing executables on Linux], page 17.

2 What's new

This version 28 February 2018

GCC:

- [CMPE200GCC-168] [CMPE200GCC-181] `__fini()` and `__init()` code contains se_illegal opcodes
- [CMPE200GCC-183] Fix use of cr7 on e_bc instructions
- [CMPE200GCC-184] Fix crxor on vle code

Binutils:

- [CMPE200GCC-176] AS generates wrong instruction opcode for instruction `wait`

Newlib:

- [CMPE200GCC-126] [CMPE200GCC-178] printf float work incorrecly for newlib_nano library
- [CMPE200GCC-170] [CMPE200GCC-180]

3 Migration to version 1.2

3.1 Update linker script

New version 1.2 of compiler have full EABI VLE compliance. For correct work of existing projects user must check linker script in his project and apply changes if necessary. Correct linker script must contain:

- KEEP for .init and .fini sections
- .ctors and .dtors sections
- .preinit_array .init_array and .fini_array sections

See necessary section in linker script example below.

```
.text :
{
  *(.text.startup)
  *(.text)
  KEEP(*(.init)
  KEEP(*(.fini)
  *(.text.*)
} > m_text

.ctors :
{
  /* gcc uses crtbegin.o to find the start of
      the constructors, so we make sure it is
      first.  Because this is a wildcard, it
      doesn't matter if the user does not
      actually link against crtbegin.o; the
      linker won't look for a file to match a
      wildcard.  The wildcard also means that it
      doesn't matter which directory crtbegin.o
      is in.  */
  KEEP (*crtbegin.o(.ctors))
  KEEP (*crtbegin?.o(.ctors))
  /* We don't want to include the .ctor section from
      the crtend.o file until after the sorted ctors.
      The .ctor section from the crtend file contains the
      end of ctors marker and it must be last */
  KEEP (*(EXCLUDE_FILE (*crtend.o *crtend?.o ) .ctors))
  KEEP (*(SORT(.ctors.*)))
  KEEP (*(.ctors))
} > m_text
.dtors :
{
  KEEP (*crtbegin.o(.dtors))
  KEEP (*crtbegin?.o(.dtors))
  KEEP (*(EXCLUDE_FILE (*crtend.o *crtend?.o ) .dtors))
  KEEP (*(SORT(.dtors.*)))
  KEEP (*(.dtors))
} > m_text

.preinit_array :
{
  PROVIDE_HIDDEN (__preinit_array_start = .);
  KEEP (*(.preinit_array))
  PROVIDE_HIDDEN (__preinit_array_end = .);
} > m_text
.init_array :
{
  PROVIDE_HIDDEN (__init_array_start = .);
  KEEP (*(SORT(.init_array.*)))
  KEEP (*(.init_array ))
  PROVIDE_HIDDEN (__init_array_end = .);
} > m_text
.fini_array :
{
  PROVIDE_HIDDEN (__fini_array_start = .);
  KEEP (*(SORT(.fini_array.*)))
  KEEP (*(.fini_array ))
  PROVIDE_HIDDEN (__fini_array_end = .);
} > m_text
```

3.2 Change instruction names

Opcodes and syntax for e_cmpwi and e_cmplwi aliases were changed. Use e_cmpl16i instead of e_cmplwi, and e_cmp16i instead of e_cmpwi.

3.3 e200z4 decorated storage instructions

Since from binutils 2.28 mainline these instructions are also supported. But assembler doesn't accept them with `-mvle` option. To enable these instructions user need to use `-me200z4` option which was intoduced in 2.28. Objdump also has `-Me200z4` switch to show mnemonics for these instructions.

4 New features

4.1 Mixed BookE and VLE code assembly and linking

Use `-mno-vle` key to force BookE codegen, even if `-mvle` key is used. For mixed VLE & BookE code corresponding linker script shall be provided.

For example:

```
.text_booke : { INPUT_SECTION_FLAGS (!SHF_PPC_VLE) *(.text*) } > text_e
.text_vle :  { INPUT_SECTION_FLAGS (SHF_PPC_VLE) *(.text*) } > m_text
```

First line will force pick non-VLE .text sections. Second line will force pick VLE .text sections.

4.2 BookE to VLE translation

Use `-Wa,-ppc_asm_to_vle` option to enable translation for assembler file. Option requires `-mvle` option. Following BookE instructions can be translated:

> addi addic addic. addis andi. andis. b bc bcl bctr bctrl bdnz bdnzl bdz bdzl beq beql bf bfl bge bgel bgt bgtl bl ble blel blr blt bltl bne bnel bng bngl bnl bnll bns bnsl bnu bnul bso bsol bt btl bun bunl clrlslwi clrlwi clrrwi cmpi cmpli cmplwi cmpwi crand crandc crclr creqv crmove crnand crnor crnot cror crorc crset crxor extlwi extrwi inslwi insrwi isync lbz lbzu lha lhau lhz lhzu li lis lmw lwz lwzu mcrf mfdec mtdec mulli nop ori oris rfi rlwimi rlwinm rlwnm rotlw rotlw. rotlwi rotrwi sc slwi srwi stb stbu sth sthu stmw stw stwu subfic subi subic subic. subis xnop xori

Follwoing VLE-compatible instructions can be translated to shorter form:

> mr or

4.3 LSP intrinsics support

In this compiler version LSP intinsic functions are introduced.

Use `-mlsp` option to enable LSP intinsic functions support in GCC and the same option for assembler.

LSP intrisic functions avaliable in the form __builtin_lsp_xxx where xxx is the name of LSP instruction:

```
out64_64 = __builtin_lsp_zaddd(in64_64, in32_32_b, in32_32_c);
out64_64 = __builtin_lsp_zldd(ptr_a, 8);
```

4.4 SPE2 intrinsics support

While SPE support implemented in mainline GCC SPE2 wasn't available at all. In this compiler version SPE2 intinsic functions are introduced.

Starting from version 1.2 of compiler SPE ABI implied by default. That means when option `-mspe` or `-msp2` is used, then `-mapi=spe` will also be set implicitly by default. To

reproduce old behavior user needs to use `-mspe2 -mabi=no-spe"`. All existing user libraries which uses SPE or SPE2 should be rebuilt in this case.

Use `-mspe2` option to enable SPE2 intinsic functions support in GCC and the same option for assembler.

SPE and SPE2 intrisic functions avaliable via API in the form:

```
o_v64 = __ev_addd(i_eva, i_evb)
```

and via builtin functions in the form:

```
o_v64 = __builtin_spe2_evaddd(i_eva, i_evb)
```

List of supported SPE2 intrinsic functions via API is given in NXP SPE2PIM Rev. 1.0-2 08/2013 document.

4.5 Generate error for e200z425 errata

There's a problem with the e200z425 cores.

The problem occurs at any page with offset 16K + 2 when a long branch with a target displacement of 0x3ffe is preceded by a 32 bit instruction. The code needs to be relinked to be located at another address.

If you want just to disable this error use option `-Wl,-no-e200z425-rel-error`

4.6 SPR registers support in GDB client

GDB client now supports SPR registers if corresponding target description (target.xml) is supplied by GDB server. SPR registers are implemented as pseudoregisters in the same manner as SPE etc.

By interface convention between S332 Design Studio for Power and GDB client following implementation is used. SPRs register numbers starts from 1000 in GDB client internal table. The offset is fixed even other pseudoregisters are available. Besides SPR registers are always hided from MI `-data-list-register-names` command result.

The table below contains list of currently supported SPR registers and their offset.

SPR name	**regnum**	**SPR name**	**regnum**	**SPR name**	**regnum**
dec	1000	ivor2	1041	mas0	1082
srr0	1001	ivor3	1042	mas1	1083
srr1	1002	ivor4	1043	mas2	1084
pid0	1003	ivor5	1044	mas3	1085
decar	1004	ivor6	1045	mas4	1086
csrr0	1005	ivor7	1046	mas6	1087
csrr1	1006	ivor8	1047	tlb0cfg	1088
dear	1007	ivor9	1048	tlb1cfg	1089
esr	1008	ivor10	1049	l1finv1	1090
ivpr	1009	ivor11	1050	hid0	1091
usprg0	1010	ivor12	1051	hid1	1092
utbl	1011	ivor13	1052	l1csr0	1093
utbu	1012	ivor14	1053	l1csr1	1094
sprg0	1013	ivor15	1054	mmucsr0	1095
sprg1	1014	spefscr	1055	bucsr	1096
sprg2	1015	l1cfg0	1056	mmucfg	1097
sprg3	1016	l1cfg1	1057	l1finv0	1098
sprg4	1017	ivor32	1058	svr	1099
sprg5	1018	ivor33	1059	tir	1100
sprg6	1019	ivor34	1060	npidr	1101
sprg7	1020	ivor35	1061	ddam	1102
tbl	1021	ctxcr	1062	dac3	1103
tbu	1022	dbcr3	1063	dac4	1104
pir	1023	dbcnt	1064	dbcr7	1105
pvr	1024	dbcr4	1065	dbcr8	1106
dbsr	1025	dbcr5	1066	ddear	1107
dbcr0	1026	iac5	1067	dvc1u	1108
dbcr1	1027	iac6	1068	dvc2u	1109
dbcr2	1028	iac7	1069	edbrac0	1110
iac1	1029	altctxcr	1070	dmemcfg0	1111
iac2	1030	iac8	1071	imemcfg0	1112
iac3	1031	dberc0	1072	devent	1113
iac4	1032	mcsrr0	1073	sir	1114
dac1	1033	mcsrr1	1074	mpu0cfg	1115
dac2	1034	mcsr	1075	mpu0csr0	1116

dvc1	1035	mcar	1076	l1csr2	1117
dvc2	1036	dsrr0	1077	usprg4	1118
tsr	1037	dsrr1	1078	usprg5	1119
tcr	1038	dbcr6	1079	usprg6	1120
ivor0	1039	sprg8	1080	usprg7	1121
ivor1	1040	sprg9	1081		

TUI Commands

`info registers all`
: Print the names and values of all registers, including SPRs.

`info registers` *`regname ...`*
: Print the *relativized* value of each specified register *regname*. Where *regname* is SPR name from table above.

```
(gdb) info registers all
...
r31            0x40003d20 1073757472
pc             0x40002c24 0x40002c24 <main+12>
msr            0x1000 4096
cr             0x20000000 536870912
lr             0x40000246 0x40000246
ctr            0x0 0
xer            0x0 0
dec            0x70600000 1885339648
srr0           0xffffc000 4294950912
srr1           0x0 0
...
(gdb)

(gdb) info registers esr
esr            0x8000020 134217760
(gdb)
```

MI Commands

Following MI commands are used to control SPR registers.

`-data-list-register-values N` *`regnum`*
: Display the registers' contents

`-data-write-register-values N` *`regnum value`*
: Write new register value.

Where *regnum* is register number from table above. For additional info about format please refer to GDB documentation.

For example to read ESR register:

```
(gdb)
-data-list-register-values N 1008
^done,register-values=[number="1008",value="134217760"]
(gdb)
```

and write new value to ESR register:

```
(gdb)
-data-write-register-values N 1008 32853
^done
(gdb)
```

5 Known issues

- [CMPE200GCC-54] Disassembler listing is corrupted. It does not match with debugger disassembly view and it's not complete.
  Workaround: usually it happens when `-S` option is used. Disable this option.

6 Release history

Version	Date	Description
1.0	27 November 2015	See Section 6.1 [1.0], page 14
1.1	8 June 2016	See Section 6.2 [1.1], page 14
1.1.1	23 June 2016	See Section 6.3 [1.1.1], page 15
1.1.2	10 August 2016	See Section 6.4 [1.1.2], page 15
1.1.3	7 September 2016	See Section 6.5 [1.1.3], page 15
1.2	27 May 2017	See Section 6.6 [1.2], page 15
171017	17 October 2017	See Section 6.7 [171017], page 16
28 February 2018	28 February 2018	See Chapter 2 [What's new], page 2

6.1 Version 1.0

6.2 Version 1.1

Fixed:

- [ENGR00377284] Compiler uses stswi/lswi instructions even if they are not supported by e200 when -Os -O3 switches are enabled.
- [ENGR00377271] Decorated storage instructions should be supported
- [ENGR00374776] Bitfield write access could be optimized to shorter data type write instr
- [ENGR00373844] Incorrect code generation for the C expression with optimization level 0.
- [ENGR00373558] Support /dev/null on Windows for Ecos config tool

Changes:

• Following instructions marked as VLE according to PowerISA 2.07b:

evlddepx evstddepx e_sc se_rfgi sld sld. srad srad. srd srd. stbcx. stbdx stddx stfddx sthcx. sthdx stvebx stvehx stvepx stvepxl stvewx stwdx vcmpbfp. subfo subfo. subfmeo subfmeo.

• Changes in accordance with EFP2_rev1.4_spec.pdf:

  – Moved opcodes for

    efdcfsid efdcfuid efdctsidz efdctuidz

  – Added missed instructions

    evfssqrt evfscfh evfscth evfsmax evfsmin evfsaddsub evfssubadd evfssum evfsdiff evfssumdiff evfsdiffsum evfsaddx evfssubx evfsaddsubx evfssubaddx evfsmulx evfsmule evfsmulo efsmax efsmin efdmadd efdmsub efdnmadd efdnmsub efssqrt efscfh efscth efdmax efdmin efdsqrt efdcfh efdcth

• Added mapping for SPE2 instructions in accordance with SPE2PIM Rev.1.0-2:

evdotphsssi evdotphsssia evdotpwsssi evdotpwsssia

• Added instructions from Engineering Bulletin EB689 Rev. 0, 2/2008 for e200z3 and e200z6 cores:

evfsmadd evfsmsub evfsnmadd evfsnmsub efsmadd efsmsub efsnmadd efsnmsub

6.3 Version 1.1.1

This version contains critical fixes for version 1.1

Fixed:

- [ENGR00379772] Compiler may generate the instruction e_subfic with invalid immediate value when size optimization is enabled.
- [ENGR00379484] Compiler assigns a wrong register (r9) when extended inline asm statement with se_bclri %0,%1 instruction is used.
  Note: User should manually set correct constraints for short VLE instructions. Added VLE constraints in documentation.

6.4 Version 1.1.2

Fixed:

- [ENGR00381415] GDB incorrectly prints full expression that this variable object represents
  Note: Utilized patch from this message in GDB mailing list.
- [ENGR00381558] GCC codegen emits illegal 'stwu' instruction in VLE mode

6.5 Version 1.1.3

Fixed:

- [ENGR00382613] Typo in extendqisi2 in md causes wrong codegen with se_extsb

6.6 Version 1.2

GCC:

- GCC base updated to 4.9.4
- Full EABI VLE compliance. User must follow Section 3.1 [Update linker script], page 3.
- Strict volatile bitfields not forced anymore by default for e200zX cores. User should use `-fstrict-volatile-bitfields` instead.
- Added qsort to libiberty to make GCC host-independent codegen.
- LSP intrinsics support
- SPE2 intrinsics support

Newlib and multilib:

- Multilib structure were optimized. From now default cpu is e200z0 and its libraries are placed in top of "lib" folder. All other cores have their libraries under subfolders (e200z2, e200z3, e200z4, e200z6, e200z7).
- Added SPE libraries for Z3, Z4 and Z6 cores.
- Added semihosting library in Newlib to support print to semihosting console.
- NewLib build with `-ffunction-sections -fdata-sections`.

GDB and binutils:

- GDB client now supports Power Architecture SPR registers.
- Binutils base updated to 2.28.

- Fixed relocation bug in binutils. Correct split16_a, split16_d type relocations. Types were reversed and bit shift on split16_d was wrong. More info in binutils-2.28 news here
- Opcodes and syntax for e_cmpwi and e_cmplwi aliases were changed. Section 3.2 [Change instruction names], page 5.
- e200z4 specific decorated storage instructions now enabled by additional switch Section 3.3 [e200z4 decorated storage instructions], page 5.

Fixed issues:

- [ENGR00381829] Unable to build VLE project if -fstack-protector option is enabled. **Note:** Error message will be generated now if -fstack-protector option is used for VLE mode.
- [CMPE200GCC-3] `-mspe2` option doesn't enable SPE intrinsics and builtin macro `__SPE__`.
- [CMPE200GCC-5] Disassembler doesn't show SPE2 instructions
- [CMPE200GCC-10] mtmas0, mtmas2, mtmas3, mtmas4 and mtmas6 instructions are missing
- [CMPE200GCC-34] Sync with CW bunch of mtSPR and mfSPR instructions
- [CMPE200GCC-35] MPU instructions (mpure, mpuwe, mpusync) are not recognized by GCC\AS
- [CMPE200GCC-52] ISEL instruction is not enabled by default for e200zX cores. Use `-mno-isel` to disable.
- ICE issues are fixed [CMPE200GCC-48] [CMPE200GCC-49] [CMPE200GCC-50] [CMPE200GCC-51] [CMPE200GCC-91] [CMPE200GCC-92] [CMPE200GCC-93] [CMPE200GCC-118]
- [CMPE200GCC-103] Incorrect optimization of bitfield
- [CMPE200GCC-125] powerpc-eabivle-gdb has stopped working
- [CMPE200GCC-132] Linker is generating bad opcode for e_add2i. instruction

6.7 Version 171017

GCC:

- ISEL disabled by deault for VLE targets. User can enable it explicitly by `-misel` option. But note that in some particular cases compilation might fail due to GCC framework and VLE ISA limitations.

GDB and binutils:

- GDB client with Python. See Chapter 8 [GDB with Python], page 18.

7 Installing executables on Linux

"32-bit compatibility libraries" should be installed just in order to run 32-bit toolchains:

in Ubuntu 14:

```
sudo apt-get install lib32z1 lib32ncurses5 lib32bz2-1.0
```

in Ubuntu 16:

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386 lib32z1
```

in Debian:

```
sudo apt-get install lib32z1 lib32ncurses5 lib32stdc++6
```

in CentOS:

```
sudo yum install glibc.i686
```

8 GDB with Python

Additionally if you want to use gdb python build (powerpc-eabivle-gdb-py) on Linux, you need to install 32 bit python2.7.

```
sudo apt-get install libpython2.7:i386
```

To use gdb python build (arm-none-eabi-gdb-py.exe) on Windows, you need to install 32 bit python2.7 no matter 32 or 64 bit Windows. Please get the package from python.org.