ARM64 CPU Feature Registers

Author: Suzuki K Poulose <suzuki.poulose@arm.com>

This file describes the ABI for exporting the AArch64 CPU ID/feature registers to userspace. The availability of this ABI is advertised via the HWCAP_CPUID in HWCAPs.

1. Motivation

The ARM architecture defines a set of feature registers, which describe the capabilities of the CPU/system. Access to these system registers is restricted from EL0 and there is no reliable way for an application to extract this information to make better decisions at runtime. There is limited information available to the application via HWCAPs, however there are some issues with their usage.

a. Any change to the HWCAPs requires an update to userspace (e.g libc) to detect the new changes, which can take a long time to appear in distributions. Exposing the registers allows applications to get the information without requiring updates to the toolchains.
b. Access to HWCAPs is sometimes limited (e.g prior to libc, or when ld is initialised at startup time).
c. HWCAPs cannot represent non-boolean information effectively. The architecture defines a canonical format for representing features in the ID registers; this is well defined and is capable of representing all valid architecture variations.

2. Requirements

a. Safety:

   Applications should be able to use the information provided by the infrastructure to run safely across the system. This has greater implications on a system with heterogeneous CPUs. The infrastructure exports a value that is safe across all the available CPU on the system.

   e.g, If at least one CPU doesn't implement CRC32 instructions, while others do, we should report that the CRC32 is not implemented. Otherwise an application could crash when scheduled on the CPU which doesn't support CRC32.

b. Security:

   Applications should only be able to receive information that is relevant to the normal operation in userspace. Hence, some of the fields are masked out(i.e, made invisible) and their values are set to indicate the feature is 'not supported'. See Section 4 for the list of visible features. Also, the kernel may manipulate the fields based on what it supports. e.g, If FP is not supported by the kernel, the values could indicate that the FP is not available (even when the CPU provides it).

c. Implementation Defined Features

   The infrastructure doesn't expose any register which is IMPLEMENTATION DEFINED as per ARMv8-A Architecture.

d. CPU Identification:

   MIDR_EL1 is exposed to help identify the processor. On a heterogeneous system, this could be racy (just like getcpu()). The process could be migrated to another CPU by the time it uses the register value, unless the CPU affinity is set. Hence, there is no guarantee that the value reflects the processor that it is currently executing on. The REVIDR is not exposed due to this constraint, as REVIDR makes sense only in conjunction with the MIDR. Alternately, MIDR_EL1 and REVIDR_EL1 are exposed via sysfs at:

```
/sys/devices/system/cpu/cpu$ID/regs/identification/
                                              \- midr
                                              \- revidr
```

3. Implementation

The infrastructure is built on the emulation of the 'MRS' instruction. Accessing a restricted system register from an application generates an exception and ends up in SIGILL being delivered to the process. The infrastructure hooks into the exception handler and emulates the operation if the source belongs to the supported system register space.

The infrastructure emulates only the following system register space:

```
Op0=3, Op1=0, CRn=0, CRm=0,4,5,6,7
```

(See Table C5-6 'System instruction encodings for non-Debug System register accesses' in ARMv8 ARM DDI 0487A.h, for the list of registers).

The following rules are applied to the value returned by the infrastructure:

a. The value of an 'IMPLEMENTATION DEFINED' field is set to 0.
b. The value of a reserved field is populated with the reserved value as defined by the architecture.
c. The value of a 'visible' field holds the system wide safe value for the particular feature (except for MIDR_EL1, see section 4).
d. All other fields (i.e, invisible fields) are set to indicate the feature is missing (as defined by the architecture).

4. List of registers with visible features

1. ID_AA64ISAR0_EL1 - Instruction Set Attribute Register 0

Name	bits	visible
RNDR	[63-60]	y
TS	[55-52]	y
FHM	[51-48]	y
DP	[47-44]	y
SM4	[43-40]	y
SM3	[39-36]	y
SHA3	[35-32]	y
RDM	[31-28]	y
ATOMICS	[23-20]	y
CRC32	[19-16]	y
SHA2	[15-12]	y
SHA1	[11-8]	y
AES	[7-4]	y

2. ID_AA64PFR0_EL1 - Processor Feature Register 0

Name	bits	visible
DIT	[51-48]	y
SVE	[35-32]	y
GIC	[27-24]	n
AdvSIMD	[23-20]	y
FP	[19-16]	y
EL3	[15-12]	n
EL2	[11-8]	n
EL1	[7-4]	n
EL0	[3-0]	n

3. ID_AA64PFR1_EL1 - Processor Feature Register 1

Name	bits	visible
MTE	[11-8]	y
SSBS	[7-4]	y
BT	[3-0]	y

4. MIDR_EL1 - Main ID Register

Name	bits	visible
Implementer	[31-24]	y
Variant	[23-20]	y
Architecture	[19-16]	y
PartNum	[15-4]	y
Revision	[3-0]	y

NOTE: The 'visible' fields of MIDR_EL1 will contain the value as available on the CPU where it is fetched and is not a system wide safe value.

5. ID_AA64ISAR1_EL1 - Instruction set attribute register 1

Name	bits	visible
I8MM	[55-52]	y
DGH	[51-48]	y
BF16	[47-44]	y
SB	[39-36]	y
FRINTTS	[35-32]	y

GPI	[31-28]	y
GPA	[27-24]	y
LRCPC	[23-20]	y
FCMA	[19-16]	y
JSCVT	[15-12]	y
API	[11-8]	y
APA	[7-4]	y
DPB	[3-0]	y

6. ID_AA64MMFR0_EL1 - Memory model feature register 0

Name	bits	visible
ECV	[63-60]	y

7. ID_AA64MMFR2_EL1 - Memory model feature register 2

Name	bits	visible
AT	[35-32]	y

8. ID_AA64ZFR0_EL1 - SVE feature ID register 0

Name	bits	visible
F64MM	[59-56]	y
F32MM	[55-52]	y
I8MM	[47-44]	y
SM4	[43-40]	y
SHA3	[35-32]	y
BF16	[23-20]	y
BitPerm	[19-16]	y
AES	[7-4]	y
SVEVer	[3-0]	y

8. ID_AA64MMFR1_EL1 - Memory model feature register 1

Name	bits	visible
AFP	[47-44]	y

9. ID_AA64ISAR2_EL1 - Instruction set attribute register 2

Name	bits	visible
RPRES	[7-4]	y

Appendix I: Example

```
/*
 * Sample program to demonstrate the MRS emulation ABI.
 *
 * Copyright (C) 2015-2016, ARM Ltd
 *
 * Author: Suzuki K Poulose <suzuki.poulose@arm.com>
 *
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 * This program is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License version 2 as
 * published by the Free Software Foundation.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 */

#include <asm/hwcap.h>
#include <stdio.h>
#include <sys/auxv.h>

#define get_cpu_ftr(id) ({                                    \
              unsigned long __val;                            \
              asm("mrs %0, "#id : "=r" (__val));              \
              printf("%-20s: 0x%016lx\n", #id, __val);       \
      })

int main(void)
{

      if (!(getauxval(AT_HWCAP) & HWCAP_CPUID)) {
              fputs("CPUID registers unavailable\n", stderr);
              return 1;
      }

      get_cpu_ftr(ID_AA64ISAR0_EL1);
      get_cpu_ftr(ID_AA64ISAR1_EL1);
      get_cpu_ftr(ID_AA64MMFR0_EL1);
      get_cpu_ftr(ID_AA64MMFR1_EL1);
      get_cpu_ftr(ID_AA64PFR0_EL1);
      get_cpu_ftr(ID_AA64PFR1_EL1);
      get_cpu_ftr(ID_AA64DFR0_EL1);
      get_cpu_ftr(ID_AA64DFR1_EL1);

      get_cpu_ftr(MIDR_EL1);
      get_cpu_ftr(MPIDR_EL1);
      get_cpu_ftr(REVIDR_EL1);

#if 0
      /* Unexposed register access causes SIGILL */
      get_cpu_ftr(ID_MMFR0_EL1);
#endif

      return 0;
}
```