Armv8-A Fact Sheet and Notes

Armv8 architecture contains three profiles: A class, R class, and M class. There are two execution states that will impact the register widths and instruction sets: `AArch64` and `AArch32`. There are three instruction sets: `AArch64` which use fixed-length instructions set of 32-bit encoding, `AArch32-A32` which use fixed-length instruction set of 32-bit encoding, `AArch32-T32` which use variable-length instruction set containing both 16-bit and 32-bit encodings.

Armv8 provides two Security states: Non-secure and Secure. Under Non-secure state, only non-secure registers/address can be accessed. Under Secure state, both non-secure and secure regsiteres/address can be accessed.

Armv8 incorporates 4 (or less, depend on implementation) different exception levels. EL0 is *unprivileged* and typically runs normal software. Exception level beyond EL0 are all privileged. EL1 typically runs (host or guest) operating system. EL2 runs host operating system or hypervisor which controls guest OS. EL3 is the highest exception level and is able to change between Non-secure security state to Secure security state. Some properties are independent among different exception level, e.g. the virtual address range of EL1 and EL0 are totally different, each with its own translation controls.

Armv8 provides two classes of register: 1 general/SIMD and FP registers and 2 system registers. System registers name will most likely to append the lowest exception level that can access the registeras a suffix. (`<register_name>.ELx`, where `x=0,1,2,3`) System registers include general system control registers, debug registers, generic timer registers, performance monitor registers, and activity monitor registers.

Application level model

The important things to notice in this sections are (regular and system) registeres, PSTATE, exception level handling, memory model.

In memory model, the important things to notice are memory address space and types, atomicity and concurrency, coherency and ordering, memory barriers, caches and synchronization, memory type and attributes

Notice that the exception level of the application level is conventionally EL0.

Registers

- There are 31 general purpose registers, `R0` to `R30`. Each register can be accessed as 64-bit from `X0` to `X30` or 32-bit from `W0` to `W30`. There is another ZR register represents 0, which doesn't have to be a physical register
- SP is a 64-bit stack pointer register. The first 32 bits can be accessed through `WSP`
- `PC` is a 64-bit program counter holding the address of the current instruction. It can only be updated on a branch, exception entry, or exception return.
- `V0-V31` are 32 SIMD and FP registers. Each register can be accessed as 128-bit with name `Q0-Q31`, or 64-bit with name `D0-D31`, or 32-bit with name `S0-S31`, or 16-bit with name `H0-H31`, or 8-bit with name `B0-B31`, or 128-bit vector of elements, or 64-bit vector of elements. The format of vectors are as follows
  - For a 128-bit vector:Vn`{.2D, .4S, .8H, .16B}`
  - For a 64-bit vector: `Vn{.1D, .2S, .4H, .8B}`
- `FPCR` and `FPSR` are 2 SIMD and FP control and status registers. The purpose of `FPCR` is to control floating-point behavior. It contains several bits that control floating-point precision and calculation behaviors. Using the `MRS` and `MSR` instruction to read/write it. `FPSR` provides floating-point system status information, which is also accessible through `MRS` and `MSR`
- PSTATE includes the current status of the PE. Condition flags(NZCV) and exception masking bits(DAIF) are the only PSTATEs that are accessible at `EL0` using `MRS` and `MSR` with `NZCV` and `DAIF` special-purpose registers. More information in the system level model section.

Software Control Features

- Exceptions cause a change of program flow and will trigger the exception handler to execute. Exceptions include interrupts, memory system aborts, `UNDEFINED` instruction execution, system calls, secure monitor or hypervisor traps, debug exceptions. More information in the system level model section.
  - `SVC` instruction causes a supervisor call exception where EL0 software can make a system call to its hosting OS
- Wait for interrupt and event: `WFI` instruction indicates no further execution is required until a WFI wake-up event occurs. `WFE` instruction indicates no further execution is required until a WFE wake-up event occurs. Both permits entry to a low-power state.
- `YIELD` provides a hint that the current thread priority is low and can potentially be yielded.
- More information about these control features in the system level model section.

Memory model

- The address calculations are always performed using 64-bit register (under AArch64). The top eight address bits can be configured to be used as a tag. If so, the bits [63:56] are not considered when determining whether the address is valid and will never propagate to the program counter.
- Address tagging is a better schema to detect memory leaks/overflow introduced in Armv8.5. More info: https://www.usenix.org/system/files/login/articles/login_summer19_03_serebryany.pdf
- There are two categories of memory: Normal type and device type. Normal memory incorporates bulk memory operations, both read/write. Device memory has additional attributes such as gathering, reordering and write acknowledgement.
  - Normal memory can be speculative. A write to a memory location with the Normal attribute completes in finite time. Accesses can be merged before tapping the target memory. Typically, access to normal memory will not cause side effects.
  - A Normal memory location has a shareability attribute that is one of: Inner Shareable, Outer Shareable, Non-shareable. Inner sharable domain is a subset of outer shareable domain. Similar property holds for cacheability.
  - Device memory access can cause side-effects, or where the value returned for a load can vary depending on the number of loads performed.
  - For device memory types, there are: `Device-nGnRnE`, `Device-nGnRE`, `Device-nGRE`, and `Device-GRE`, order from stronger to weaker memory type. Because device memory will cause side-effects, speculative data accesses are not allowed.
  - Gathering refers to whether it is permitted to either merge multiple memory accesses of the same type, read or write, to the same memory location into a single transaction, or merge multiple memory accesses of the same type, read or write, to different memory locations into a single memory transaction on an interconnect.

    - Reordering indicates that accesses to the location can be reordered within the same rules that apply to accesses to Normal Non-cachable memory.
    - Early Write Acknowledgement: setting nE means that only the endpoint of the write access returns a write acknowledgement of the access, and no earlier point in the memory system returns a write acknowledgement.
    - Memory attributes are controlled by priviledged software. Mismatched memory attributes occur when one of the memory type, shareability and cacheability differs.
- Atomicity is a memory feature. There are two types of atomicity: single-copy atomicity and multi-copy atomicity.
- Cache is high speed memory with size limitation due to its high cost. To balance between cache size and cost, L1/L2 cache is implemented. Memory has cacheability and shareability attributes that can be divided into inner, outer, or none.
- Endianness refers to the order in which consecutive bytes are ordered. A64 instructions have a fixed length of 32 bits and are always little-endian. Instructions such as `REV32` are used to convert endianness for data in a register. All memory-mapped peripherals defined in the Arm architecture must be little-endian.

System Level Architecture

The important points to notice here are exception levels, execution state, security state, virtualization, process state and reset, exception entry and return, synchronous and asynchronous exceptions, system calls and wait-for-interrupt-or-event low-power state, system level memory model, virtual memory system architecture, memory tagging,

Exception levels, Execution state and Security state

- EL0 is unpriviledged, EL1/2/3 are priviledged. EL2 provides support for virtualization, EL3 provides support for switching between two Security states
- All implementations must include EL0 and EL1. EL2 and 3 are optional.
- On taking an exception, the EL can only increase or remain the same. On returning from an exception, the EL can only decrease or remain the same. EL0 cannot be a target exception level.
- Typical exception level usage model: EL0 for applications, EL1 for OS kernel and associated functions that are typically described as priviledged, EL2 for the hypervisor and EL3 for the secure monitor.
- Precise exception: able to know which line caused the exception. Other than SError interrupt, all exceptions taken to AArch64 is required to be precise.
- Synchronous exception is generated as a result of direct execution or attempted execution of an instruction, and the return address presented to the exception handler is guaranteed to indicate the instruction that caused the exception, and the exception is previse. On the contrary, asynchronous exception is NOT generated as a result of direct execution or attempted execution of the instruction stream, and the return address is not guaranteed to indicate the instruction that caused the exception, and the exception is imprecise.

    - Synchronous exceptions include: execution of an instruction that is UNDEFINED. (e.g. executing instructions at an inappropriate exception level, execute instructions when they are disabled), misaligned SP/PC, exception generating instructions `SVC`, `HVC`, or `SMC`, data aborts,
    - Asynchronous exceptions are known as interrupts. There are two different kinds of interrupts: physical interrupts and virutal interrupts. More on this later
- There are 2 security state: secure state and non-secure state. Secure state is able to visit both secure and non-secure physical address space. Non-secure

Virtualization

- EL2 provides a set of features that support virtualizing an Armv8-A implementation. It includes a hypervisor running in EL2 that is responsible for switching between virutal machines(EL1/EL0), a number of guest OS(EL1), and for each guest OS, applications are in EL0
- Virtual interrupts, including vSError, vIRQ and vFIQ, are implemented when EL2 is implemented. All virtual interrupts are always taken to EL1 and can only be taken from EL0/1

PSTATE

- `ELR` registers hold preferred exception return address.
- PSTATE includes all of the following: condition flags(NZCV), execution state controls, exception mask bits, access control bits, timing control bits, and speculation control bits
- Accessing PSTATE fields uses `MRS` and `MSR` instructions and special-purpose registers
- Saved Program Status Registers are used ot save PE state on taking exceptions
- Cold reset and warm reset

Exception entry and returns

- On taking an exception to AArch64 state: the PE state is saved in the SPSR regsiter at target exception level. The preferred address is saved in the ELR register. PSTATE.{D, A, I, F} are set to 1. If the exception is synchronous or SError, information regarding the reason of the exception is saved in ESR at target exception level. SP will be selected for the target exception level, etc.
- ESR records the information regarding the exception: EC[31:26] holds the exception class field indicating the cause of exception. IL[25] length bit indicates whether a trapped insutrction was a 16-bit or a 32-bit instruction. ISS[24:0] instruction specific syndrome field.
- On returning from an exception, it will be either to a previosly executing thread or entry to a new execution thread (initialization of a hypervisor by a Secure monitor, initialization of an operating system by a hypervisor, application entry from an operating system or hypervisor). The PC and PSTATE will be restored.

System calls and wait-for-event/interrupt

- System calls are triggered by `SVC`, `HVC`, or `SMC` instructions.
- The instrution `WFE` and `WFET` and `WFI` and `WFIT` will let PE enter a low-power state
- The Event Register for a PE is set by `SEV` , `SEVL`, or an exception return, etc
- Attempts to enter a low-power state made by software executing at EL0-2 migh tbe trapped to a higher exception level

System level memory model

- Armv8-A includes a virtual memory system architecture. More on this later.
- Device memory are outer shareable and non-cacheable. Normal memory attribute could be any type. (e.g. inner sharable/non-sharable, write-through cacheable, etc)
- Point of coherency: the point at which all agetns are guaranteed to see the same copy of a memory location for accesses, normally is the main memory
- Point of Unification: the PoU for a PE is the point by which the instruction and data caches and the translation table walks of the PE are guaranteed to see the same copy of a memory
- Point of Persistence: the point at or beyond PoC where a write to memoyr is maintained when system power is removed. It recovers when power is restored to the affected locations in memory
- There is a series of cache clean/invalidate operations that might include the concepts above, e.g. `IC IALLU`, `DC IVAC`, etc

Armv8-A Virtual Memory System Architecture(VMSA)

- Provides MMU to translate virtual addresses to physical addresses. Might involve a single stage or two sequential stages of translation
- The translations are defined in dependently for different exception levels and security states
- VMSAv8 refers to the overall translation scheme, within which an address translation has one or two stages. VMSAv8-32/64 is the translation scheme for a single stage of address translation
- Virtual address is an address used in an instruction, as a data or instruction address. All addresses held in PC, LR, SP, or ELR are virtual addresses
- In AArch64 state, the virtual address has a maximum address width of one of the following: 48-bits normally, or 52 bits wen FEAT_LVA is implemented and the 64 KB translation granule is used.
- A stage of address translation can support one or two VA ranges.
    - For translation supporting only a single VA range, a 48-bit VA width gives a VA range of 0x0000000000000000 to 0x0000FFFFFFFFFFFF. A 52-bit VA width will give the maximum of 0x000FFFFFFFFFFFFF.
    - For translation supporting two VA subranges, one at the bottom of the full 64-bit address range, and one at the top, the scheme is as follows: The bottom VA range runs up from address 0x0000000000000000 to 0x0000FFFFFFFFFFFF (for 52-bit just another F). The top VA subrange runs up to address 0xFFFFFFFFFFFFFFFF, from 0xFFFF000000000000
    - A 48-bit VA range corresponds to an address space of 256TB. A 52-bit VA range corresponds to an address space of 4PB
    - Translation scheme can be configured to include fewer addresses
- In translation regimes that provide two stages of address translation, there is an Intermediate Physcial Address. Output address from the stage 1 translation or the Input Address for the stage 2 translation is the Intermediate Physical Address (IPA). In a translation regime with only a single stage, the IPA is the same as PA.
    - For a virtual address accessed in Secure state, it can be translated to either the Secure or the None-secure PA space
    - When in Non-secure state, a VA is always mapped to the Non-secure PA space.
- Physical address is the address of a location in a physical memory map. Similar policy follows for physical address.
- Address tagging (not memory tagging) utilizes the top 8 bits of the VA to determine whether the address is out of range and hence causes a Translation fault, and whether the address requries invalidation when performing a TLB invalidation instruction by address.
- Pointer authentication is used to support authentication of the contents of a register before that register is used as the target of an indirect branch, or as load.
    - Pointer authentication provides an instruction that inserts a Pointer authentication Code(PAC) into the upper bits of a register. The bits used are the extension bits that do not hold valid address bits. The inserted PAC value is calculated from the value of the register and one other 64-bit value.
    - It also provides an instruction that extracts the PAC from the upper bits of a register and checks whether the value is correct, based on the value of the register and one other 64-bit value. If the value is correct, replaces the PAC with the extension bits. Otherwise, replaces the PAC with the extesion bits, except that two bits of the extension are set to a fixed unique number. This will cause a translation fault because the VA is not mapped.
    - It also provides an instruction to remove the PAC, replacing it with extension bits, without any verification
    - Relevant instructions: `PACIASP`, `PACIAZ`, etc. for adding PAC in front of the pointer. `AUTIASP`, `AUTIAZ` for authenticating PAC for the pointer.
    - The bit fields vary depending on whether the address tagging is used
- The address translation system
    - The Memory Management Unit (MMU) controls address translation, memory access permissions, and memory attribute determination and checking, for memoyr accesses made by the PE
    - The MMU will take an input address and either return an associated output address and the memory attributes of that address, or is unable to perform the translation and thus causes an exception to be generated. This type of exception is called an MMU fault. System registers are used to report any MMU faults
    - The translation granule specifies the granularity of the mapping from input address to output address. It defines both the page size for a stage of address translation, where a page is the smallest block of memory for which an input address to output address mapping can be specified, as well as the size of a complete translation table for that stage of address translation.
    - Translation table entries can be cached in a Translation Lookaside Buffer(TLB). The TLB will define: for accesses made from secure state, whether the access is to the Secure or Non-secure address map, the memory access permissions, and the memory region attributes
    - As discussed before, the translation regime is either
        - A single stage of address translation, from VA to PA
        - Two sequeal stages of address translation: stage 1 from VA to IPA and stage 2 from IPA to PA

      The translation regimes are controlled by which exception level it is enabled/at currently
    - An MMU fault is described as either a stage 1 MMU fault or a stage 2 MMU fault
    - For a single stage of address translation, a Translation Table Base Register indicates the start of the first translation table required for a mapping from input address to output address.
    - The VMSAv8-64 translation provides
        - Up to 4 levels of address lookup
        - Translation granule size of 4KB, 16KB or 64KB
        - 52-bit if conditions are met, otherwise 48-bit
    - Things like PA ranges can be configured by bit fields in system registers
    - The memory translation granule size defines both the maximum size of a single translation table and the memory page size. It defines the number of address bits required to address a memory page and the number of bits that can be resolved in a single translation table lookup
    - There is an instruaction AT that can be used to translate virutal addresses
- VMSAv8-64 memory aborts

    - There are four ways to cause an exception on a failed memory access: debug exception, alignment fault, MMU fault, and external abort
    - Fault Address Register and Exception Syndrome Regsiters will contain relevant information
    - MMU faults are synchronous exceptions as data aborts or instruction aborts, it includes
        - Alignment fault
        - Permission fault
        - Translation fault
        - Address size fault
        - Synchronous external abort on a translation table walk
        - Access flag fault
        - TLB conflict abort
- Translation lookaside buffers (TLBs)
    - TLB reduces the average cost of a memory access by chaching the results of translation table walks. TLBs behave as caches of the translation table information,