Lecture 01.03 A CPU programming model

A *central processing unit* (CPU) has three functions that are repeated endlessly: **central processing unit**

1. fetch an instruction from memory,
2. translate the instruction, and
3. execute it.

Typically, the *control unit* of a CPU fetches instructions from memory and translates them. It then sends to the *datapath* of the CPU to be processed. It does this by means of *registers*, which are small, special purpose units of memory in the datapath. **control unit** **datapath** **registers**

01.03.1 Core ARM registers

A developer of an embedded system with a given CPU must understand it at the "application-level," which is distinguished from the "system-level" of the operating system. An application-level "view" of the ARM processor registers has thirteen *general-purpose 32-bit registers* named R0–R12 and three *special-purpose registers* named SP, LR, and PC (also called R13–R15) (ARM, 2014, p. A2-45). **general-purpose registers** **special-purpose registers**

The *stack pointer* SP (R13) register contains the *memory address* of, and therefore *points* to, the top of the *active stack*. A stack holds data, such as "automatic variables," temporarily. We'll talk more about stacks, later. **stack pointer** **active stack**

The *return link* LR (R14) register is used, for instance, to store the current memory address of the calling program during a subroutine call. **return link**

The *program counter* PC (R15) register contains the memory address of the current instruction plus eight (bytes)—that is, of two instructions-from-now.[4] **program counter**

The general-purpose registers typically hold data, such as `int`s, `double`s, and `char`s.

01.03.2 Other ARM registers

The 32-bit *application status register* (APSR) stores the program's last-executed instruction return status in flags: **application status register**

- N: negative condition (e.g. two's complement negative MSbit),
- Z: zero condition (e.g. equal from comparison),

[4] For an interesting discussion of "why the offset," see this informative SO answer.

REVIEW DRAFT

- C: carry condition (e.g. unsigned overflow from addition),
- V: overflow condition (e.g. signed overflow from addition), and
- Q: overflow or saturation condition (e.g. from DSP)

encoded as single bits. These flags can be tested by the next instruction for conditional execution. A nibble of the APSR stores the GE: greater-than or equal flag.

execution state registers

The *execution state registers* allows special instruction sets, such as Thumb, to be executed; contains special Thumb instructions; and sets the register endianness mapping (big-endian or little-endian).

vector floating-point unit

extension registers

The Xilinx Z-7010 Coretex-A9 has the optional ARMv7-A *vector floating-point unit* VFPv3 ISA extension, which enables high-performance and efficiency of floating-point arithmetic. The extension has its own, dedicated *extension registers*.

01.03.3 Types of instructions

Below are some examples of the types of instructions a CPU might encounter:

- load or store (to/from CPU registers),
- transfers (between registers),
- move (memory→memory),
- set/reset bits,
- shift/rotate,
- arithmetic (add, subtract, multiply, divide, negate),
- logic (ands, ors, etc.),
- conditional branches and jumps,
- unconditional branches and jumps, and
- subroutines.

01.03.4 Addressing modes

addressing mode

Addressing modes specify how the CPU is to calculate the memory address for a load or a store operation. For the ARMv7-A ISA, the address is composed of two parts: a *base register* value and an *offset* (ARM, 2014, p. A4-176). The base register can be any core ARM register. The offset must have one of the following three formats.

base register

offset

Immediate An unsigned number, it can be summed with (or subtracted from) the value of the base register.

REVIEW DRAFT

Register A value from a core ARM register other than PC.

Scaled register A shifted value from a core ARM register other than PC summed with (or subtracted from) the value of the base register.

These lead to the following three addressing modes:

Offset The offset is summed with (or subtracted from) the base register, forming the memory address.

Pre-indexed Same as "Offset," followed by the new address is then assigned to the base register.

Post-indexed The memory address is the value of the base register. Then the base register is offset.

REVIEW DRAFT