ARMv8 - 64-bit Architecture

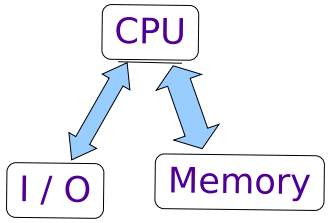
First look at the arm64 architecture and assembly language

Assembly Language and Architecture

- Assembly-language programs consist of instructions that manipulate processor hardware elements
  - Goal of manipulations: implement high-level-language program behaviors
- Processor designs determine what instructions are available
- Understanding assembly language requires understanding the processor architecture

Computer Organization – Programmer's View

- Fundamental components:
  - CPU
  - Memory
  - I/O ("everything else")



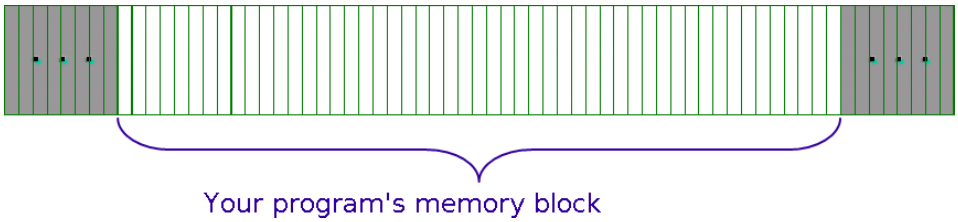
- Possible connection schemes:
  - Single Controller *(above right)*
  - Direct Memory Access (DMA) *(below)*

Memory Organization

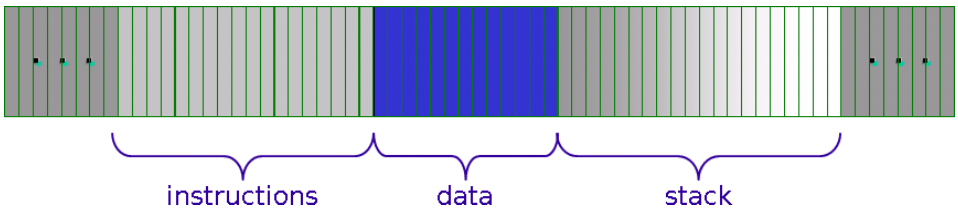
- The memory subsystem is a 1-dimensional array of storage locations
  - Each location has a unique address (or "array index")
  - Each location is made up of m bits (m >= 1)
  - Location's contents is any one of 2^m bit patterns
- Memory can hold:
  - Programs: bit patterns represent instructions
  - Data: bit patterns represent numbers, characters, etc.
- Volatile contents: R/W memory, or RAM
  - DRAM is common for main memory
- Nonvolatile (unchangeable): Read-Only, or ROM
  - PROM, EEPROM, Flash – erasable/reprogrammable ROM

How a program occupies memory

- When a program runs it occupies a block of RAM

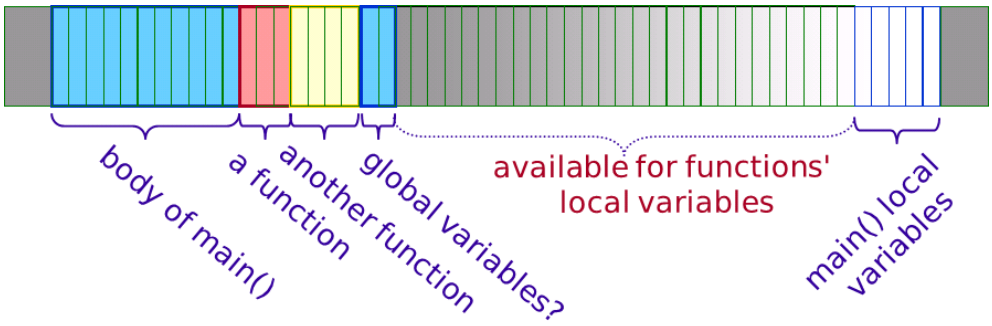


- The block is divided into instructions, data, and stack segments, each in part of the memory



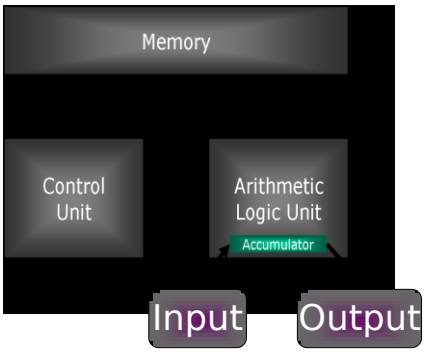
Dividing up the program memory

- Each function, including main(), has its own unchanging portion of the instruction memory
  - Any global variables occupy memory similar to instruction memory.
- Function-local variables exist in the stack only while the function is executing



the Central Processing Unit

- a.k.a. CPU or Processor
  - (or microprocessor)
- Reads program instructions
- Accesses program data
- Performs numerical and logical operations
- von Neumann model
  - basis for most modern computers



John von Neumann's IAS Processor

- Developed at the Institute for Advanced Studies, Princeton, NJ
- Single, shared memory for program instructions and data

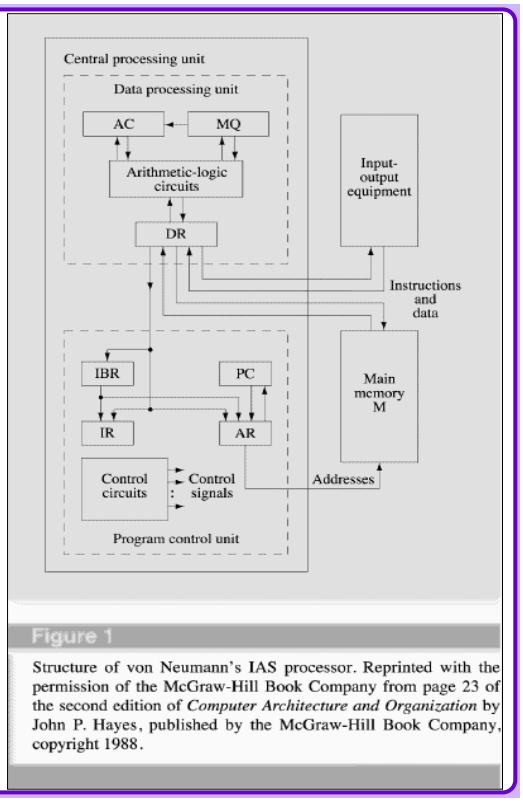
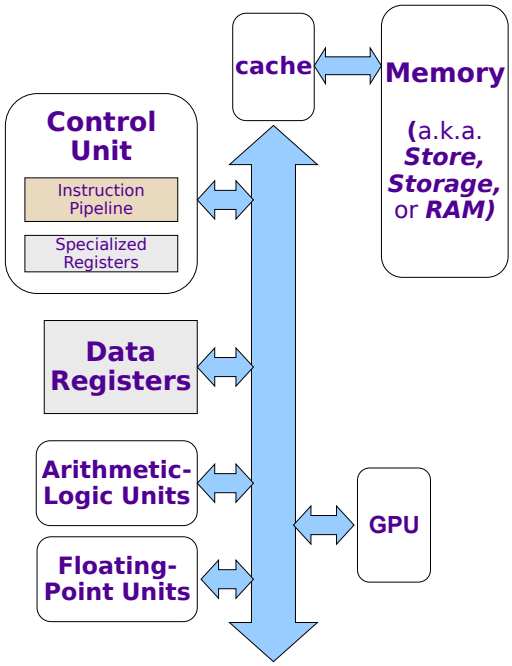


Figure 1

Structure of von Neumann's IAS processor. Reprinted with the permission of the McGraw-Hill Book Company from page 23 of the second edition of *Computer Architecture and Organization* by John P. Hayes, published by the McGraw-Hill Book Company, copyright 1988.

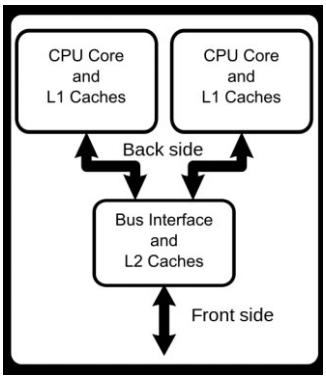
Modern CPU elements

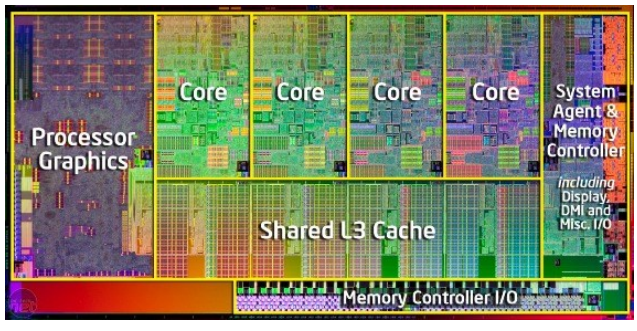
- Control Unit
  - Instruction pipeline
  - Specialized registers
    - contain *execution state*
    - control program flow
- General-Purpose Registers
  - Hold data values for ALU, FPU
- ALU, FPU, GPU
  - Do the computations
- Cache
  - Improves memory accesses
  - Generally not visible to programmer
- Internal buses
  - Connect things



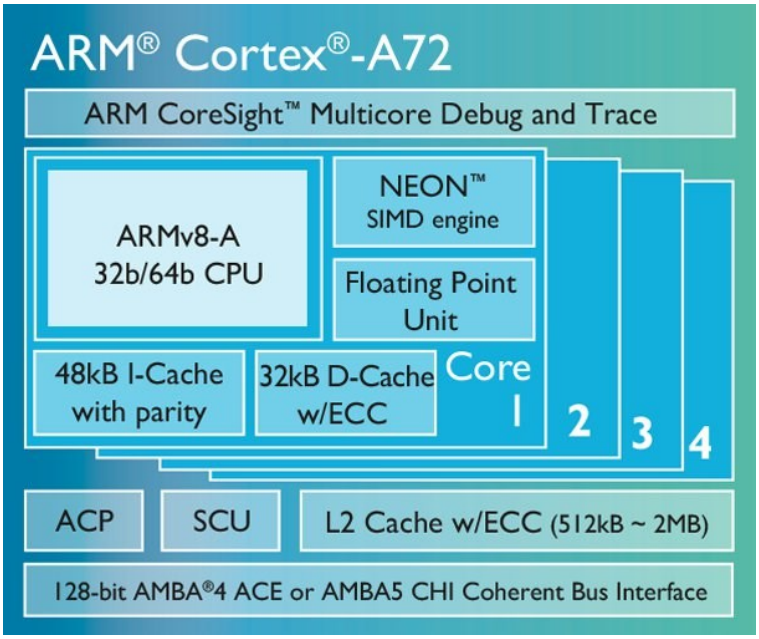
Multi-core Processors

- Each CPU is a "core"
- One chip holds 2, 4, 8, ... cores
- Cores share memory
- Cores may have local L1 cache, may share L2 and L3 cache
- A CPU with a GPU is a form of *heterogeneous* multicore processor



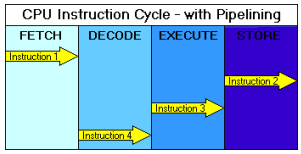


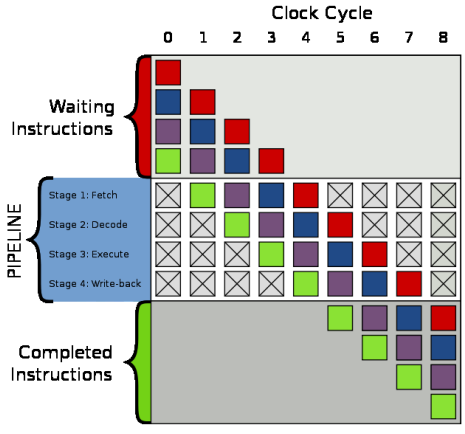
Raspberry Pi 4B Processor Block Diagram



Pipelined Designs

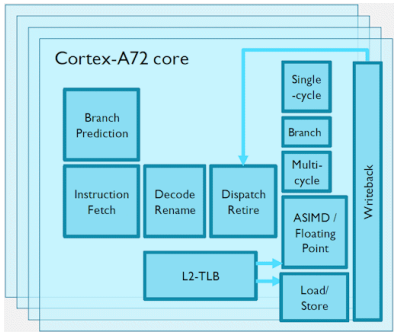
- Separate circuits perform each stage of the fetch-execute cycle
- In a *pipelined* design the separate circuits work simultaneously, on sequential instructions

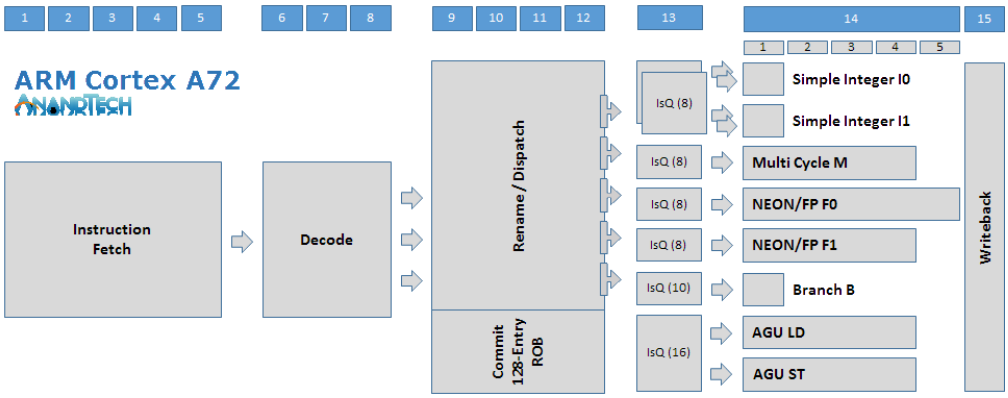




RPi Instruction Pipeline

- *Right*: simplified
- *Below*: showing clock cycles



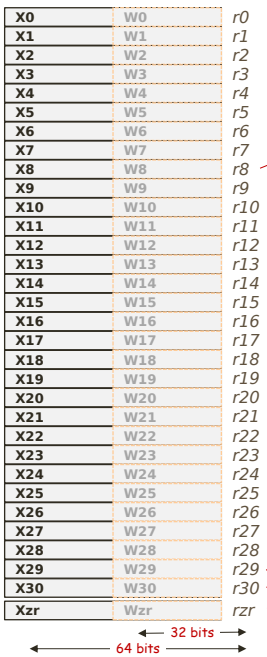


The ARM Architectural Models

- Aarch32 (armhf, armel)
  - Original models, optional floating-point
  - Provides 32-bit registers and instructions
- T32 (Thumb2)
  - 16-bit instructions expand to 32 bits on use
  - Compatible with Aarch32
- Aarch64 (arm64)
  - 64-bit registers, some new instructions
  - Floating-point, SIMD hardware standard
  - Context switch between Aarch64 , Aarch32

ARMv8 General-Purpose Register Set

- r0 - r7
  - Arguments, return values
- r8
  - Syscall number / Returned-struct pointer
- r9 - r15
  - Temporary
- r16 - r18
  - Intra-procedure/platform
  - Avoid using these
- r19 - r28
  - Callee preserves these
- r29, r30
  - Frame, link registers:
  - Not general-purpose



Syscall number / Return pointer

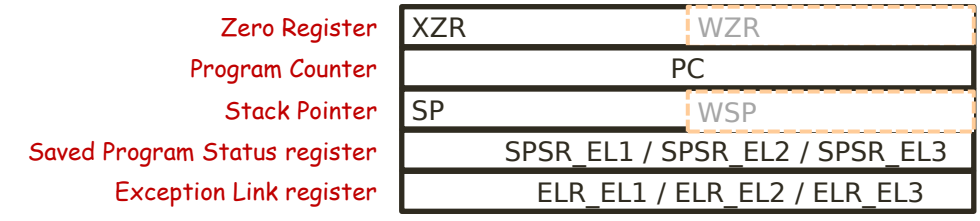
Register names:
- X-names
  - 64-bit versions
- W-names
  - Lower 32 bits
- r-names
  - generic references

Frame Pointer

Link Register

Zero Register / Stack Pointer

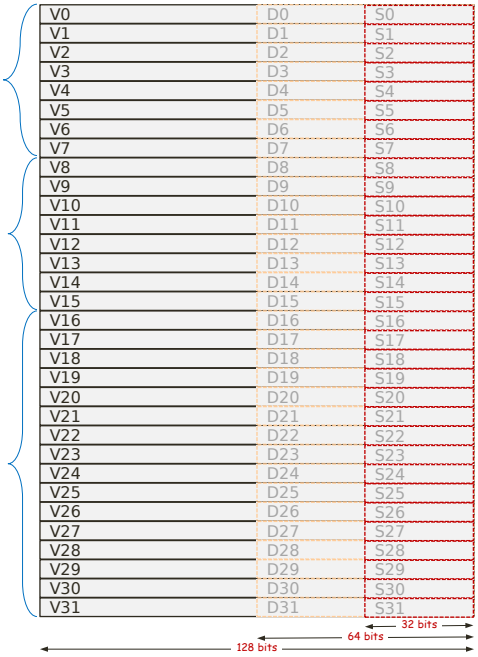
Aarch64 Special Registers



- Zero Register
  - reads always return 0
  - writes are always discarded
- PC
  - no explicit access
- SP, SPSR, EL registers:
  - Separate set for each Exception level EL0 - EL 3
- SPSR
  - includes NZCV condition-code flags
- EL
  - holds exception return address
- *Some instructions treat ZR or SP as "r31"*

Aarch64 VFP/SIMD (NEON) Register Set

- V0 - V7
  - Arguments, return values
- V8 – V15
  - Callee-saved (lower 64 bits only)
- V16 – V31
  - Spare temporary registers
    - Not in Aarch32
- D0 – D31
  - Double-precision registers
    - D0 - D15 also in Aarch32
- S0 - S31
  - Single-precision registers
    - S0 - S15 also in Aarch32



Exception Levels

- ARMv8 security model
- "Userland" operates at Level 0 (L0)
  - Least privilege
  - Library Functions
- Operating system operates at L1
  - System calls
- Virtualization runs at L2
- Hardware has highest privilege level, L3