VON NEUMAN

Memory holds both data and instructions, CPU fetches instruction from memory; Program memory is read only, data is w/r therefore program contents can't be modified; Must coordinate data and instruction memory access.

|  |  |  |  |
| --- | --- | --- | --- |
| MEM | <---- Addr ---->  <---- Data ----> | PC |  |
| CPU | |
| IR |  |

HARVARD

2 separate memories; allows 2 simultaneous memory fetches; allows for greater memory bandwidth; Both memories are w/r

|  |  |  |  |
| --- | --- | --- | --- |
| Data Memory | <---- Addr ---->  <---- Data ----> | PC |  |
| CPU | |
|  |  |
| Program Memory | <---- Addr ---->  <---- Data ----> |
| IR |  |

Memory is bottleneck for system - request, then read. More MB per second.

But Von Neuman - extreme low power cost and cost.

**CISC -** computes a task in as few assembly code lines as possible = 1 complex instruction; HW must understand that 1 instruction is actually series of operations, &1 CISC instruction = multiple clock cycle.

=> fewer memory accesses & code density

**RISC** - Use simple instructions that can be ex in 1 cc pipeline (stage)

=> greater number of instructions = more memory BUT simple instructions = better pipelining for greater performance = greater HW utilization.

|  |  |
| --- | --- |
| CISC | RISC |
| Emphasis on HW | Emphasis on SW |
| Multi cc/complex inst | Single cc/inst |
| Mem-to-mem exe  - Microcode = reg to reg | Reg-to-reg exe |
| Smaller code size | Larger code size |

Haswell = more stages

CPU all stages to execute an instruction, how to optimize theory

=> CortexM3 is a 32b low power microP. 100Mhz, 64KB RAM, 512KB ROM.  
=> 3 Stage pipeline

|  |  |  |
| --- | --- | --- |
|  | ID | EXE |
| IF | Thumb Decomp/ARM decode | RegRead| Shift/ALU | Reg Write |
| Fetches 1 32b  or 2 16b  - IF queue which buffers additional insts | Decodes inst & places addr on reg file for read | Reg read, processes & writesback to reg |

ALU Asynch

Latency => time it takes Inst to get through pipeline. Throughput => # of inst exe / time period

pipelining has greater throughput w/o reduction of latency

NVIC (nested vectore) => Interrupt Controller

Memory Protection Unit (MPU) = invokes rules for accessing memory, used so that untrusted user programs may not access kernel regions or other priviledged memory sections.

SYSTICK = count down timer, used to generate certain interrupts (ie periodic ISR, Multitasking etc.)

WIC = unit for waking up CPU

ROM = small lookup table that stores config mem (mem map of system devices)

BUSMATRIX = interconnect used to transfer data on different busses simultaneously (one master/transaction)

REST ARE DEBUGGING = SW-PP (serial wire debug port), DWT (Data watch point), ETM, ITM etc.

Arm uses the advanced Microcontroller Bus Arch (AMBA) - open standard on-chip interconnect specification (ie. protocols, handshakes etc.)

Cortex M3 uses the following AMBA busses:  
1) Advanced Peripheral Bus (APB) - reduceBW, used for peripheral transactions  
2) ARM high-Perf BUS (AHB) - greaterBW/datawidth transfers, use bursts.

INTERRUPUTS: signal send to the processor signaling that an event has generated which requires immediate attention. CPU Suspends current task, Saves state, executes interrupt handler (ISR) to deal w/ event. once finished, the CPU restores its state and continues to execute task.

ARM INTERRUPT SYSTEM/EXEPTION HANDLING  
-Rerrered to as NVIC  
-UPON ISR entry, CPU pushes RO-R3, & special registers R12-R15 on stack. these are popped off when exiting ISR

NVIC posses the Following features:  
- Vectored Interupt support - ISR addr are stored in a vector table in memory, therefore interupt looks up ISR addr in a table vs using SW to branch and locate ISR  
 - data bus - stacking/saving state, inst bus - access ISR addr.

- Nested Interrupt support - ext & internal interrupts/exception can be programmed  
 - if an interrupt w/ greater level is triggered during lower level ISR, new ISR will override the interrupt and execution. lower level interrupt goes into a pending state.

DYNAMIC PRIORITY CHANGES - int may be greater priority level dynamically so that their ISRs are not interrupted by another interrupt. (max 256 levels)

TAIL CHAINING - Improves interrupt latency during nested interrupt.

polling always checks.