Department of Electronic and Telecommunication Engineering

University of Moratuwa, Sri Lanka

EN2030 - Fundamentals of Computer Organization and Design

PROCESSOR DISSECTION REPORT

Group 32

Submitted by

SIRITHUNGA M.R.A.	180609B
SOMARATHNE P.M.P.H.	180616T
THALAGALA B.P.	180631J

Submitted on

November 17, 2020

Contents

Contribution from each member to the project

Index	**Name**	**Contribution(Sections Covered)**
180609B	SIRITHUNGA M.R.A.	* Micro-Architecture (Data Path and Controller) * ALU functions
180616T	SOMARATHNE P.M.P.H.	* Instruction Set * Instruction classes and Instruction Format
180631J	THALAGALA B.P.	* Cache Memory and Memory Interfacing * Timing related to Memory

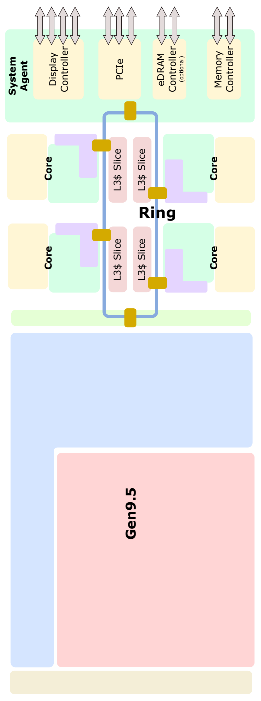
Table 1: Contribution from each member to the project

PDF is clickable

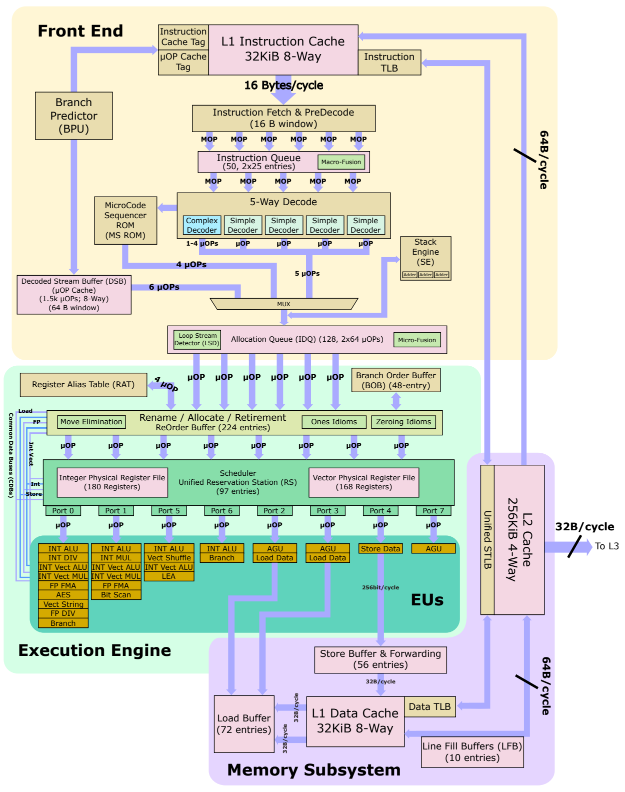
1 Introduction

1.1 Intel Core i3-8300

Core i3-8300 is a desktop processor released in Q'2 of 2018 by Intel Corporation. It is based on Coffee Lake microarchitecture and manufactured using 14nm process. This processor is developed as a low-end performance processor with four cores, 4 threads, 3.7GHz base frequency, 8MB Intel Smart Cache and 62 W Thermal Design Power(TDP)[9].



(a) Quad Core Core-i3 8300 Processor



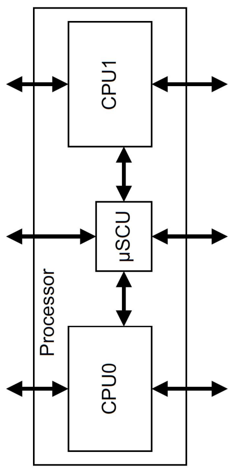
Front End
Instruction Cache Tag
µOP Cache Tag
L1 Instruction Cache 32KiB 8-Way
Instruction TLB
16 Bytes/cycle
Branch Predictor (BPU)
Instruction Fetch & PreDecode (16 B window)
Instruction Queue (50, 2x25 entries)
Macro-Fusion
MicroCode Sequencer ROM (MS ROM)
5-Way Decode
Complex Decoder
Simple Decoder
Stack Engine (SE)
Decoded Stream Buffer (DSB) (µOP Cache) (1.5k µOPs; 8-Way) (64 B window)
6 µOPs
4 µOPs
5 µOPs
MUX
Loop Stream Detector (LSD)
Allocation Queue (IDQ) (128, 2x64 µOPs)
Micro-Fusion
Register Alias Table (RAT)
Branch Order Buffer (BOB) (48-entry)
Move Elimination
Rename / Allocate / Retirement ReOrder Buffer (224 entries)
Ones Idioms
Zeroing Idioms
Integer Physical Register File (180 Registers)
Scheduler Unified Reservation Station (RS) (97 entries)
Vector Physical Register File (168 Registers)
Port 0
Port 1
Port 5
Port 6
Port 2
Port 3
Port 4
Port 7
INT ALU
INT DIV
INT Vect ALU
INT Vect MUL
FP FMA
AES
Vect String
FP DIV
Branch
INT MUL
Bit Scan
Vect Shuffle
LEA
AGU
Load Data
Store Data
EUs
Execution Engine
64B/cycle
Unified STLB
L2 Cache 256KiB 4-Way
32B/cycle
To L3
Store Buffer & Forwarding (56 entries)
Load Buffer (72 entries)
L1 Data Cache 32KiB 8-Way
Data TLB
Line Fill Buffers (LFB) (10 entries)
Memory Subsystem

(b) A Single Core

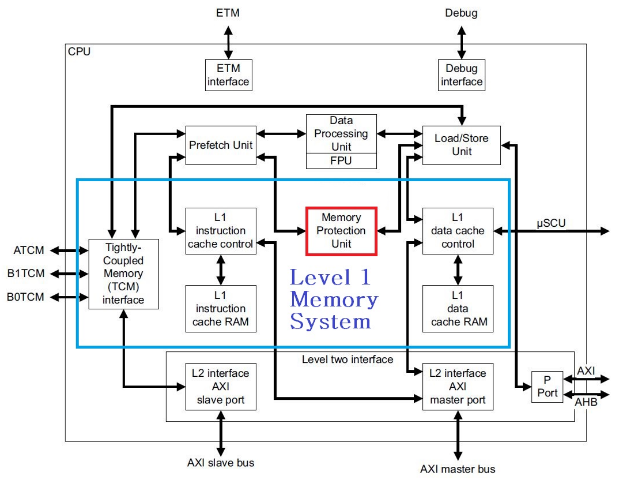
Figure 1: Intel core-i3 8300 Processor

1.2 ARM Cortex R5

Cortex-R5 is a 32-bit processor by Arm Holdings, optimized for embedded real-time mission critical applications. The processor was introduced in 2011.[15]



(a) Dual Core - Cortex -R5 Processor



(b) A Single CPU

Figure 2: ARM Cortex R5 Processor

2 Instruction Set Architecture

2.1 i3-8300: ISA

Intel i3-8300 utilizes the x86-64 ISA developed by Intel. It is a 64-bit extension to the x86 architecture. This architecture has 64-bit wide registers, which support 32-bit operations as well. There are 16 general-purpose registers and several special purpose registers including RIP, RFLAGS, XMM & MMX registers. RFLAGS is a 64-bit wide flags register. Common flags include Carry(`CF`), Parity(`PF`), Zero(`ZF`) and Overflow(`OF`). The instructions can be 1 to 15 bytes long. The source operands can be stored in registers, memory or instruction itself (immediate), while registers or memory can be used as the destination operands. All operands of an instruction cannot be in memory simultaneously.

Addressing Modes Supported[10]

1. **Immediate** - `$immediate_operand`
2. **Register** - `register`
3. **Memory direct** – `memory_address`
4. **Memory indirect** – `(register)`
5. **Base displacement** – `imm(register)`
6. **Scaled index** – `(reg1,reg2,scale)` memory address = reg1 + reg2*scale
7. **Scaled index with displacement** – `imm(reg1,reg2,scale)` memory address = imm + reg1 + reg2*scale

Some of the commonly used Instructions in x86-64[7]

`nop`	No operation	`neg`	Negate
`mov`	Move to/from mem/reg	`and/or/not/xor`	Bitwise operations
`xchg`	Exchange	`shr/sar`	Shift right logical/arithmetic
`bswap`	Byte swap	`shl/sal`	Shift left logical/arithmetic
`push/pop`	Stack usage	`cmp`	Compare
`add/adc`	Add/with carry	`jmp`	Unconditional jump
`sub/sbc`	Subtract/with carry	`je/jne/jc/jnc`	Jump if equal/ not equal/ carry/ not carry
`mul/imul`	Multiply/unsigned	`call/ret`	Call subroutine/ return
`div/idiv`	Divide/unsigned	`loop`	Loops
`inc/dec`	Increment/decrement		

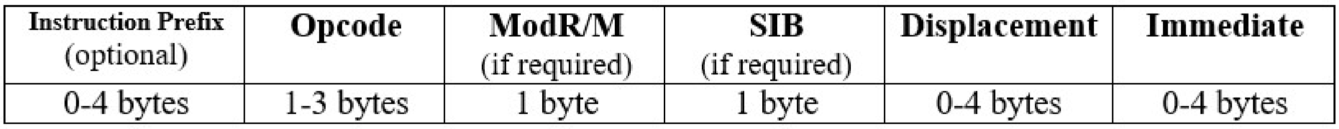
2.2 i3-8300: Instruction Classes

There are four classes of instructions in the x86-64 architecture: Data transfer instructions, Arithmetic and logic operations, Control flow instructions and Privileged (System) instructions.

1. Data transfer instructions
Data transfer between memory and registers is carried out by the `mov[size] src, dst` instruction. The size of the moved data is given by the suffix of the instruction which can be `b`(byte), `w`(word), `d`(double word) or `q`(quad word). There are few extended versions of `mov` instruction, which support features like zero-extension or sign extension during transfer.
2. ALU operations
ALU supports wide functionality and all the operations have the following general format.
`opcode[size] src2, src1/dst` ; src1 = operation(src1,src2) or dst = operation(src2)
Furthermore, Load Effective Address (`lea`) instruction can be used to utilize the address calculating hardware for normal arithmetic operations.
3. Control flow operations
Control flow instructions include unconditional and conditional branches and subroutine calls.
Unconditional branch: `jmp`
Conditional branch: `je, jne, jl, jge`
Subroutine calls: `call, ret`
4. Privileged instructions
These instructions are only accessible to the operating system and privileged software.

2.3 i3-8300: Instruction Format

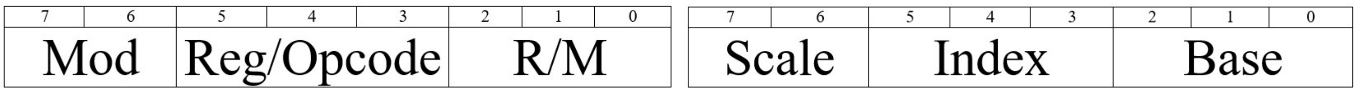
The general format of a x86-64 instruction is as follows and contains some optional fields which enhance the functionality of the instruction set. This can be considered as a variable length instruction set.



Instruction Prefix (optional)	Opcode	ModR/M (if required)	SIB (if required)	Displacement	Immediate
0-4 bytes	1-3 bytes	1 byte	1 byte	0-4 bytes	0-4 bytes

Figure 3: *Allocation of bytes within an instruction*[7]

The ModR/M byte and SIB byte together give addressing information when used.



7	6	5	4	3	2	1	0
Mod		Reg/Opcode			R/M		

7	6	5	4	3	2	1	0
Scale		Index			Base		

Figure 4: *Bit allocation within ModR/M byte and SIB byte*[7]

2.3.1 Instruction prefixes[10]

Any of the four instruction prefixes can be used to modify the instruction for extended functionality.

1. Lock and repeat prefixes, BND prefix
2. Branch hints, segment override prefixes(`CS, SS, DS, ES, FS, GS`)
3. Operand size override prefixes
4. Address size override prefixes

2.4 Cortex R5: ISA

Cortex-R5 utilizes the ARMv7-R architecture with Thumb-2 technology. This is a 32-bit RISC architecture. Instructions can be either 16 or 32-bits in length and almost all the instructions support conditional execution. The ISA can be considered as a load/store architecture as other operations only support operands from registers. There are 16-general purpose registers of which R13, R14 and R15 are used as stack pointer(SP), link register(LR) and program counter(PC), respectively. There is also a current program status register(CPSR) and a saved program status register(SPSR), which is only accessible to privileged software. The CPSR contains flag bits as well as some other fields. Coprocessor 15(CP15) is part of the processor that is responsible for control and error correction tasks. Only privileged software can access its commands and it has 16 32-bit primary registers for its operation.

2.5 Cortex R5: Instruction Classes

The instructions can be broadly categorized into 5 categories; ALU operations, data transfer operations, branch operations, SIMD/saturated operations and miscellaneous instructions.[2]

1. ALU & register operations

ALU operations can perform only on the registers or immediate values. The general format of an operation of this class is, `Operation[cond][s] Rd, Rn, Operand2` where the `cond` and `s` can be used for conditional execution and signed arithmetic respectively. The `Rd` and `Rn` must be registers while `Operand2` can be flexibly used for registers, immediate values or omitted. These operations also support special multiplication operations for efficient MAC ops.

`ADC`	Add with carry	`ADD`	Add
`MOV`	Move between regs	`MVN`	Move NOT
`SUB`	Subtract	`RSB`	Reverse Subtract
`AND`	AND	`BIC`	Bit clear
`EOR`	Exclusive OR	`ORR`	OR
`CMP`	Compare	`CMN`	Compare NOT

2. Data transfer operations

These instructions are useful for loading data from memory(`LDR`) and storing back to memory(`STR`). ISA supports four addressing modes: *Register addressing, Pre-index addressing, Pre-index addressing with write-back and Post-index addressing with write-back.* Load and store instructions have several versions to allow features including transfering a byte, multiple transfer etc. Multiple transfer instructions are available for faster transfer to/from successive memory locations.

3. Branch operations
There are multiple ways to branch in ARMv7-R including `B` for relative branching, `BL` for linked branching(during subroutines) or modifying the PC register. Static and dynamic branch prediction are supported by the ISA, along with a return stack consisting of 4-entry LIFO buffer. Direct and indirect branches can be used where indirect branching results in lesser accuracy for branch prediction.

4. SIMD/saturation operations
The 32-bit width of the system can be effectively used by packing,unpacking and operating on 8-bit or 16-bit data using SIMD commands. Saturation arithmetic is also supported for more accurate calculations during overflow conditions.

5. Miscellaneous instructions
These instructions are for general maintenance of the processor, co-processor, cache preload etc.

`CDP`	Initiate co-processor data processing operation
`MRC/MCR`	Move to/from ARM register and co-processor register
`LDC/STC`	Load/store co-processor register
`MRRC/MCCR`	Multiple register transfers
`LDCL/STCL`	Multiple load/store for co-processor

2.6 Cortex R5: Instruction Format

The ARMv7 instruction format follows a general allocation scheme. The [31:28] bits are for conditional execution while [27:25] bits select the class of the instruction. Each class has its unique bit arrangement and it is further specialized for each instruction depending on the operands. It can be seen that the register addresses and immediate values occur mostly at the same location throughout the instruction set to allow simpler decoding hardware.

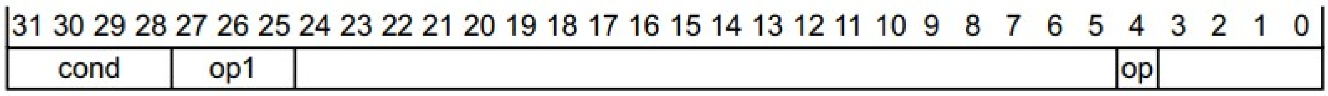


Figure 5: *General format of an ARMv7-R instruction*[1]

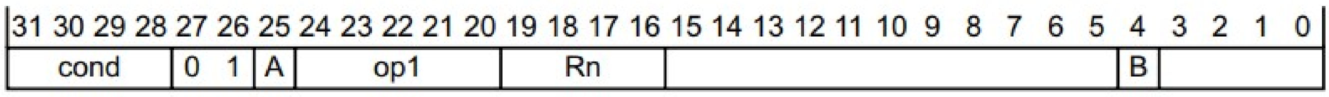


Figure 6: *General bit allocation for data transfer instructions in ARMv7-R*[1]

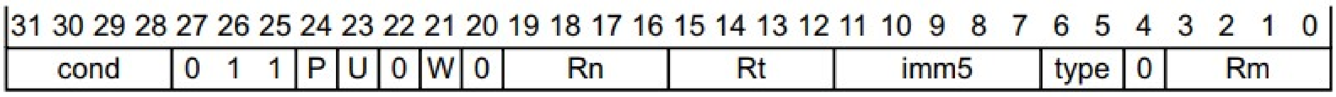


Figure 7: *Example: Bit allocation for store* `STR(register)` *instruction in ARMv7-R*[1]

3 Micro-Architecture (Data Path and the Controller)

Computer architecture consists of two main branches called instruction set architecture (ISA) and microarchitecture (μarch). A given ISA can be implemented with different microarchitectures. So, microarchitecture depends on the ISA and the technologies used in implementations. The microarchitecture is the digital logic that defines the way of how the instructions should be executed. Here we have focused on the organization of the Data path and the controller of the internal processor design[6].

3.1 Arm Cortex-R5 Processor

This is a mid-range CPU which is widely used in embedded, real-time systems. To that, ARMv7-R architecture is implemented in cortex-R5.ARM is a RISC architecture. Microcontroller Bus architecture has been embedded with it for better performance. When it comes to the internal design of the processor, the processor may have single or dual CPU configurations. However, the CPU data path is common at all 32bits data paths will be used[16].(Refer figure 2(b))

The PreFetch Unit (PFU) fetches memory card instructions, predicts branches, and forwards the Data Processing Unit (DPU) instructions. Both directions are executed, and data memory is passed using the Load / Store Unit (LSU). The PFU and LSU interfaces support the L1 interface, which contains L1 instructions and caches, and TCM interfaces. In exchange, the L1 caches are linked to the L2 memory system and the LSU connects to the L2 memory system through a more direct peripheral port. Data Store interfaces L1 to the μSCU[17] for cache management to be compatible with ACP transactions.

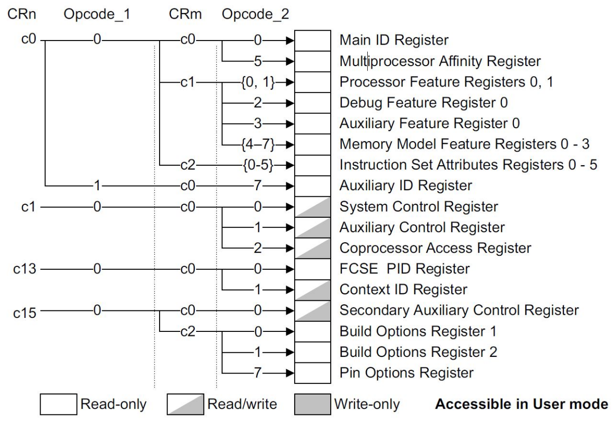


Figure 8: System control and configuration registers

There is a system control coprocessor as the controller in cortex-R5 implementation.The control signals manipulating registers, device operations management of cache and the functionality of the memory system are induced by this. There are 18 read-only registers and 7 read/write registers among them as shown here.

3.2 Intel Core i3-8300 Processor

The internal design of an intel core i3 8300 is very advanced and complicated. It is CISC based microarchitecture called 'Coffee Lake'. The actual size of a data path is 64 bits. Three levels of caches have been implemented inside it for better performance. Uni-processor configuration is used with four CPU cores and threads in the design. This is pipelined. The focus of the pipelining is to reduce power consumption and enhance overall performance. The coffee lake's pipeline is the same as intel's sky lake pipelining strategies. So, it is embedded in additional parallelism for achieving its performance. The pipeline can be split into three zones: the ***front-end***, ***execution***, and the ***subsystem for memory***(refer figure 1(b)).

By encoding instructions, the front end is supposed to feed the back end with an appropriate stream of operations that it collects from memory. There are two major routes to the front-end: the micro-operations(μOPs) cache route and the legacy path. The legacy path is the regular path by which x86 instructions of variable length are extracted, queued, and finally decoded into shorter fixed-length μOPs from the Level 1 instruction cache. The alternative and much more fitting solution is the cache route of μOPs in which a cache containing decoded μOPs receives a hit that explicitly passes the μOPs to the decode list.

The reorder buffer is visited at the back end by a micro-operation. This is where the register is reserved, renamed, and deleted. At this stage, several other optimizations are also carried out. The μOPs are forwarded to the unified scheduler from the reorder buffer. There are a few escape ports in the scheduler, each of which is connected to a set of distinct execution modules. Some units can perform basic ALU operations, others can multiply and divide, with some units, such as independent vector operations, capable of more complicated operations. The scheduler is responsible for the queuing of the μOPs to the relevant port so that the appropriate system will execute them. Both μOPs deal with memory access, load & store access. Those that can perform the memory operation would be sent to the dedicated scheduling ports. Store operations go to the buffer of the store, which is also able to forward if required. Mostly, load operations come from a load buffer. According to that, the internal design consists of three major sub designs. These are focused on the following improvements,

1. **Front end**

   I. Increase the delivery of the legacy pipeline to 5 micro operations.
   II. Increase the spread of IDQ to 6 micro-operations.
   III. Support for a 2.28x wider 64/thread allocation queue.
   IV. Improves the branch prediction unit's output.

2. **Execution engine**

   I. Increase the buffer for re-order to 224 entries.
   II. Increase the scheduler to 97 entries and to 180 entries for the integer registry file.
   III. Increase the buffer in the store to 56 entries.

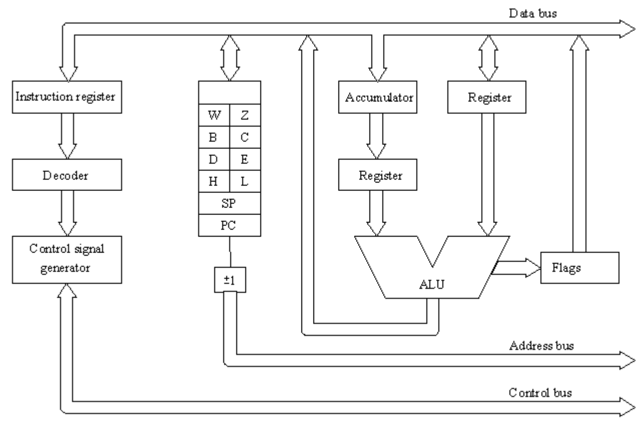
3. **Memory subsystem**

   I. Associative mapping, 8-way to 4-way set.

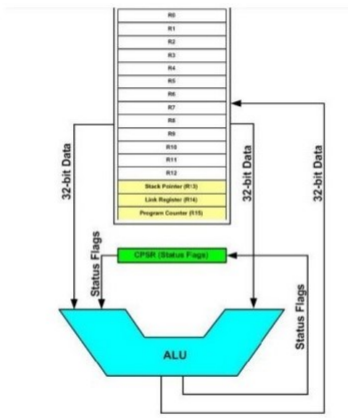
The fetching and decoding of the instructions are happening separately. Fetched instructions are stored in a queue (FIFO). After decoding, these are executed in the execution engine.The control signals control each of these three subsystems. Generating control signals are done in a control store[4].

4 ALU functions

The arithmetic and logic unit is abbreviated as ALU in the computer organization. ALU is responsible for all arithmetic and logical operations. It shall be carried out using the digital logic required for the respective set of operations.



(a) Core-i3 8300 Processor's ALU



(b) Cortex-R5 processors's ALU

Figure 9: ALUs of the two processors

4.1 Arm Cortex-R5 Processor

In the eight-stage pipeline, an ALU performs its task. It two data paths that are directly connected with 32 bits registers. Cortex-R5 is well designed for digital signal processing and efficiently performed floating-point number calculations.(figure 9(b)) Four flags stand for,

- N - Negative
- Z - Zero the ALU output subjected to NOR operation
- C - Carry the Count bit
- V - Overflow V bit output

4.2 Intel Core i3-8300 Processor

Intel's Coffee Lake architecture is used 32bits registers. So, for that ALU should be capable of manipulating these numbers in it. This is achieved with digital logic. (figure 9(a)) Two of them are embedded in a processor for,

- Physical address calculations
- Mathematical operations

Core i3 processors are widely used in desktop and laptop general-purpose computers. The frequency of the clock is about 3.7 GHz, so the ALU operates at a higher speed than its clock time.[14].

---

5 Cache Memory and Memory Interfacing

5.1 Intel Core i3-8300 Processor

5.1.1 Memory Hierarchy

Cache Memory Intel core i3-8300 processor's cache memory is organized using ***Intel's Smart Cache*** technology. That is the Last Level Cache(LLC) is shared across all cores while lower level caches are separately allocated for each core such that those are used by the respective cores privately. The shared cache allows any of the four cores to access the entire storage area of the shared cache(*in this case 8MiB LLC*) and therefore not limited to a dedicated portion of it. This leads to a number of benefits such as, increased resource utilization through providing all of the shared cache to the active cores if the other cores are in the idle mood, reduced front-side bus traffic since shared data can be fetched directly from the LLC(*if available*) into the cores rather than going all the way to the primary memory[13].

The following table summarizes the properties of each cache. Refer the *memory subsystem* of the figure 1(b) to identify the organization of core-private caches while the shared cache(LLC/L3$) is depicted inside the *Ring* of the figure 1(a).

Consider these abbreviations to refer tables, D-*Data*, I-*Instruction*, WB-*Write-back*, WT-*Write-through* U-*Unified*, S-*Shared*, SA-*Set Associative*

Cache	Mapping Technology	Cache Size	No: of Sets	Cache line size	Writing Policy
L0 µOP Cache	8-way SA	1,536 µOPs	32	6-µOP	N/A
L1 I Cache	8-way SA	32 KiB	64	64 B	N/A
L1 D Cache	8-way SA	32 KiB	64	64 B	WB
L2 U cache	4-way SA	256 KiB	1024	64 B	WB
L3 U, S Cache/LLC	Up to 16-way SA	8 MiB	8192	64 B	WB

Table 2: Cache Memory Organization in Intel core i3-8300 Processors[4]

Primary/Physical/Main Memory The next lower level memory after the L3 cache(LLC) is the System DRAM(Dynamic Random Access Memory) which is also known as the Primary/Main/Physical memory. Intel processors come in 4 different memory channel configurations as, *Single channel, Dual channels, Triple channels and Flex mode.* Intel core i3-8300 Processor is a Dual channel and has the capability of reading from or writing to the primary memory in a maximum rate of 37.5GB/s. Moreover the processor supports up to 64GB of DDR4-2400 RAMs(the 4th generation of Double Data Rate(DDR) RAMs with 2400 Mbps data transfer rate) which is also an ECC(Error-Correcting Code) memory with the ability of detecting and correcting of common types of internal data corruptions.

5.1.2 Translation-Lookaside Buffers(TLBs)

In a ***Virtual Memory System Architecture***(VMSA)(the technique of using primary memory as a cache for the secondary memory) the processor generates *virtual addresses* while the memory is accessed using the *physical addresses.* The mechanism of translating a virtual address to a physical address is called ***Address Translation/ Mapping*** and it consumes time. Because a single memory access in such a virtual system is actually a two physical memory accesses: first access to obtain the physical address corresponding to the virtual address from the *page* *table*(part of the memory where physical addresses corresponding to virtual addresses are stored) and the second access to obtain the required data stored in that physical address.

To reduce this latency, an address-translation cache which is known as ***Translation-Lookaside Buffer(TLB)*** where recent address translations are stored is used. In a single core of the Intel core i3-8300 Processor there are three TLBs and their properties are given in the following table while the physical placement is shown in the *Memory Subsystem* and the upper part of the *Front End* of the figure 1(b).

TLB	Mapping Technology	Page size	No: of Entries	Partitioning Method
I-TLB	8-way SA	4 KiB	128	Dynamic
D-TLB	4-way SA	4 KiB	64	Fixed
STLB U	12-way SA	4 KiB + 2 MiB	1536	Fixed

Table 3: TLBs Organization in Intel core i3-8300 Processors

5.1.3 Store Buffers

Each core of an Intel Core-i3 processor consists of a Store Buffer, which is located between the *Port 4*(dedicated port for storing data) of the scheduler and the *L1-Data cache* as shown in the figure 1(b). Every ***memory write*** operation carried out by the processor is temporarily stored in this buffer before they are executed. So the processor does not have to wait until the operation is finished and it can carry out the rest of the instructions freely. This mechanism increases the processor's overall performance through ***eliminating the unwanted idling***.

5.2 Arm Cortex-R5 Processor

5.2.1 Memory Hierarchy

Cache Memory Arm Cortex-R5 Processor's CPUs only have Level 1(L1) integrated cache controllers while ARM-L2 cache controllers can be connected outside of the processor instance according to the requirement, by the system designer. L1 caches are split as *L1 Instruction cache* and *L1 Data cache* in order to increase the performance through increasing the bandwidth. Moreover, their sizes can be independently configured to be between 4KB and 64KB and each cache can be disabled independently. Data and Instructions are fetched in to the L1 caches via the *AXI master port* at Level 2 Interface(shown in figure 2(b)) from the external memory.

When considering the cache organization of L1 caches, they are always implemented using ***Set Associative Mapping Technology*** in order to reduce the cache thrashing(loosing of data in a cache line at a given index, when replacing it with a new cache line with the same index) come across in the Direct Mapping Technology. One cache line consists of ***8-words*** and ***Pseudo-random cache replacement policy*** is used to *randomly select* a cache line to be replaced in a given *set*(a group of cache lines with the same index) for an incoming new cache line at the occurrence of a *cache miss*. Moreover, the ***critical word is first filled***(the word requested by the processor) to the cache line at such a cache miss, rather than waiting for the whole memory block in order to increase performance. Additionally, the writing policy can be configured using the Memory Protection Unit(MPU) to be ***either write-back or write-through***.

Tightly-Coupled Memories (TCMs) *Unpredictable access time* at a processor request is a common issue related to cache memories because, access times at a *cache hit* and a *cache miss* are different in nature. Tightly-Coupled Memories (TCMs) address this issue and provide ***low-latency*** memory access and ***consistent access time*** which is ideal for storing time-critical routines[3]. The Cortex-R5 processor can be configured to have one or two TCMs(ATCM & BTCM) which are located separately from the processor. They may contain any mix of Data and Instructions and can have capacities up to 8MB. These TCMs can be implemented as SRAMs or ROMs and typically fetch data in a single cycle. ATCM has a single port while BTCM has two ports which can be accessed simultaneously as it is implemented as two banks of RAMs. TCMs can be loaded via the *AXI slave port* at Level 2 Interface(shown in figure 2(b)) and processor can fetch instructions from the TCMs directly without going all the way to an external memory. Although the TCM is implemented as a separate RAM it does not have a dedicated

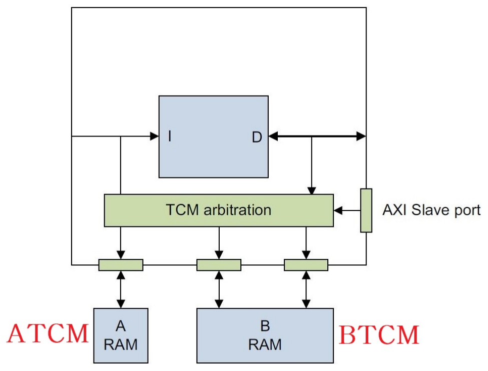


Figure 10: TCMs in Cortex-R5

address space(set of addresses) and it simply ***uses a portion of the 32-bit address space*** which is used by the processor at normal operation. Therefore when these TCMs are enabled anything at the same address space(as that is allocated for the TCMs) in the external memory is not accessible to the processor[5].

5.2.2 Memory Protection Unit(MPU)

In a multitasking system, the task currently being executed must not affect the system resources(code, data) of the other tasks. This protection mechanism is controlled by the Operating System(OS), typically with the help of both hardware and software. In ARM Cortex-R series processors which implement the ARM ***Protected Memory System Architecture***(PMSA) this feature is provided by a dedicated hardware called *Memory Protection Unit*(MPU)[3] located inside the *Level 1 Memory System*(shown in figure 2(b)) of the cores. Using the MPU, memory can be partitioned into ***zero, 12 or 16 regions*** and protection attributes can be set for each region independently. The size of such a region is specified by a 5-bit value which encodes a range of values from 32-Bytes(cache-line length) to 4GB.[5]

---

6 Timing related to Memory

6.1 Intel Core i3-8300 Processor

In coffee lake μarchitecture the processor is divided into several ***Clock Domains*** where each part maintains a different clock frequency which is applicable only to that part. All of these frequencies are some multiple of what is known as *Base Clock* which acts only as a reference for other clock domains[11],[12]. Cache memories, their related clock domains, latency and bandwidths are as follows. In addition to that system's DRAMs are operated under the *Memory clock* domain and are capable of transferring data at the rate of 8 bytes per cycle per channel[4](Core i3-8300 Processor is dual channel as mentioned previously).

Cache	Clock Domain	Fastest Latency (cycles)	Peak Bandwidth (bytes/cyc)	Sustained Bandwidth (bytes/cyc)
L1 I Cache	Core Clock	N/A	N/A	N/A
L1 D Cache	Core Clock	4	96	81
L2 U cache	Core Clock	12	64	29
L3 U, S Cache/LLC	Ring Clock	44	32	18

Table 4: Timing Related to Cache Memory in Intel core i3-8300 Processors[8]

6.2 Arm Cortex-R5 Processor

In Arm Cortex-R5 Processors there is ***only one clock*** input(***CLKIN***) for the entire CPU and the *Advanced Microcontroller Bus Architecture*(AMBA) system which consists of AXI master, AXI slave, ACP(Accelerator Coherency Port) and some other ports, is synchronized with this clock. That is even though AMBA system's clock has lower frequency, its rising edge is synchronous to the CLKIN. As mentioned previously, AXI master is used for instruction fetching and data access while AXI slave is used for external access to TCMs. Since ARM Coretx-R5 implements the both *arm* and *thumb-2* ISAs, in ARM state the memory system can supply up to two instructions per cycle while in thumb state the memory system can supply up to four instructions per cycle. In addition to that, access to external memory will take tens or even hundreds of core cycles in a normal ARM processor.

---

7 Comparison

The two processors have some significant differences. Intel Core i3-8300 is a low-performance processor designed for general purpose computing while the ARM Cortex-R5 is a mid-performance processor specialized for real-time embedded applications and digital signal processing tasks. Apart from that the underlying ISA and micro-architecture also has differences as they follow RISC or CISC principles. Cortex-R5 has an implementation that is closer to hardware resulting in higher clock speeds, mostly single clock-cycle operations, shorter and less-vivid instructions and a simpler assembly language. Core-i3 processor has a vast instruction set that has advanced functionality where some instructions can take several clock cycles to execute.

Cortex-R5 uses armv-7r architecture which has native 32bits data paths, favoring four-byte operations, while intel i3 8300 is using Intel's coffee lake architecture which uses 64bits data paths. Armv-7r is a reduced instruction set computing (RISC) based digital logic design and the i3 8300 is complex instruction computing (CISC) based. Simple instructions are operated only on registers and there is a lot of general-purpose registers in cortex-R5. Armv-7r is allowable only read and store with the main memory. The silicon cost is very low, and the power consumption is also very low. The number of general-purpose registers is less in i3 8300 processor and directly operate with main memory as well, while complex instructions are executed in a single clock cycle. There is significant amount of digital logics to manipulate such behavior. The ALU is directly connected with its 32bits registers and all logical and mathematical operations are manipulated in Cortex-R5. The ALU of i3 8300 is capable of manipulating 64bits numbers and digital logics. Like other RISC based architectures, this is explicitly fast and widely used in embedded processors. There is a well-known saying that intel architecture is like an American muscle car. It costs more and power-hungry.

Given tow processors implement two distinct memory system architectures which leads them to have two different memory hierarchies. Intel Core i3-8300 processors implement ***Virtual Memory System Architecture***(VMSA) with *Translation-Lookaside Buffers*(TLBs) while Arm Cortex-R5 processors implement ***Protected Memory System Architecture***(PMSA) which is based on a *Memory Protection Unit*(MPU). Additionally there are strong differences in cache memory organization as well. In Intel Core i3-8300 processor there are 4 levels of cache while there is only L1 cache in the Arm Cortex-R5 processor since L2 cache is optional. The rest of the main differences are summarized in the following table.

Consider these abbreviations to refer table, D-*Data*, I-*Instruction*, WB-*Write-back*, WT-*Write-through* U-*Unified*, S-*Shared*, SA-*Set Associative*

Feature	**ARM cortex-R5**	**Intel Core-i3-8300**
Primary focus	Embedded applications	General purpose
ISA category	RISC	CISC
Instruction length	Fixed length	Variable length
		Includes optional bytes
Instruction complexity	Simple instructions	Complex instructions
	Close to hardware	Enhanced functionality
Addressing modes supported	Simple Modes	Complex Modes
	Register direct and displacement	Include bases, scales and offsets
Micro Architecture	ARMv7r	Coffee Lake(8th Gen: Intel)
Digital logic design	Comparably simple	Complex
Silicon cost	Low	High
Operation	Only on registers	On registers & Directly on memory
Data paths	32 bits	64 bits
Number of Cores	Single/Dual(Configurable)	4
Memory Sys: Archi:	Protected Mem: Sys: Archi:(PMSA)	Virtual Mem: Sys: Archi:(VMSA)
TLBs	N/A	I-TLB, D-TLB, STLB-U
Cache Levels	L1-I, L1-D, Optional L2	L0-μOP, L1-I, L1-D, L2-U, L3-U-S
Cache Mapping Technology	Set Associative	Set Associative
Cache writing Policy	WB or WT (Configurable through MPU)	WB
Cache Line Size	32 Bytes	64 Bytes
TCMs	Single Port ATCM, Dual Port BTCM	N/A
Primary(physical) DRAM	4GB	Upto 64GB of DDR4-2400
ECC on Memories	Available	Available
Clock Domains	Single Clock Input(CLKIN)	Multiple Clock Domains

Table 5: Key differences in the two processors

Bibliography

[1] Arm architecture reference manual(armv7-a and armv7-r edition). https://developer.arm.com/documentation/ddi0406/latest/.

[2] Arm cortex-r programmers guide. https://developer.arm.com/documentation/den0042/a/.

[3] Arm cortex-r series programmer's guide. https://developer.arm.com/documentation/den0042/a/.

[4] Coffee Lake - Microarchitectures - Intel - WikiChip. https://en.wikichip.org/wiki/intel/microarchitectures/coffee_lake.

[5] Cortex-r5 technical reference manual. https://developer.arm.com/documentation/ddi0460/d.

[6] Definition of microarchitecture. https://www.pcmag.com/encyclopedia/term/microarchitecture.

[7] Intel® 64 and IA-32 Architectures Developer's Manual: Vol. 2A. https://www.intel.com/content/www/us/en/architecture-and-technology/64-ia-32-architectures-software-developer-vol-2a-manual.html.

[8] Intel® 64 and IA-32 Architectures Optimization Reference Manual. https://www.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-optimization-reference-manual.html.

[9] Intel® Core™ i3-8300 Processor (8M Cache, 3.70 GHz) Product Specifications. https://ark.intel.com/content/www/us/en/ark/products/129942/intel-core-i3-8300-processor-8m-cache-3-70-ghz.html.

[10] Introduction to x64 Assembly. https://www.intel.com/content/www/us/en/develop/articles/introduction-to-x64-assembly.html.

[11] Kaby Lake - Microarchitectures - Intel - WikiChip. https://en.wikichip.org/wiki/intel/microarchitectures/kaby_lake.

[12] Skylake (client) - Microarchitectures - Intel - WikiChip. https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client).

[13] Software Techniques for Shared-Cache Multi-Core Systems. https://www.intel.com/content/www/us/en/develop/articles/software-techniques-for-shared-cache-multi-core-systems.html.

[14] How Microprocessors Work. https://computer.howstuffworks.com/microprocessor.htm, April 2000.

[15] ARM Cortex-R. https://en.wikipedia.org/w/index.php?title=ARM_Cortex-R&oldid=977947427, September 2020. Page Version ID: 977947427.

[16] X. Iturbe, B. Venu, E. Ozer, and S. Das. A Triple Core Lock-Step (TCLS) ARM® Cortex®-R5 Processor for Safety-Critical and Ultra-Reliable Applications. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pages 246–249, June 2016.

[17] Arm Ltd. Cortex-R5. https://developer.arm.com/ip-products/processors/cortex-r/cortex-r5.