Module 2

Embedded Processors and Memory

Lesson 8

General Purpose Processors - I

In this lesson the student will learn the following

Architecture of a General Purpose Processor
Various Labels of Pipelines
Basic Idea on Different Execution Units
Branch Prediction

Pre-requisite

Digital Electronics

8.1 Introduction

The first single chip microprocessor came in 1971 by Intel Corporation. It was called Intel 4004 and that was the first single chip CPU ever built. We can say that was the first general purpose processor. Now the term microprocessor and processor are synonymous. The 4004 was a 4-bit processor, capable of addressing 1K data memory and 4K program memory. It was meant to be used for a simple calculator. The 4004 had 46 instructions, using only 2,300 transistors in a 16-pin DIP. It ran at a clock rate of 740kHz (eight clock cycles per CPU cycle of 10.8 microseconds). In 1975, Motorola introduced the 6800, a chip with 78 instructions and probably the first microprocessor with an *index register*. In 1979, Motorola introduced the 68000. With internal 32-bit registers and a 32-bit address space, its bus was still 16 bits due to hardware prices. On the other hand in 1976, Intel designed 8085 with more instructions to enable/disable three added interrupt pins (and the serial I/O pins). They also simplified hardware so that it used only +5V power, and added clock-generator and bus-controller circuits on the chip. In 1978, Intel introduced the 8086, a 16-bit processor which gave rise to the x86 architecture. It did not contain floating-point instructions. In 1980 the company released the 8087, the first math co-processor they'd developed. Next came the 8088, the processor for the first IBM PC. Even though IBM engineers at the time wanted to use the Motorola 68000 in the PC, the company already had the rights to produce the 8086 line (by trading rights to Intel for its bubble memory) and it could use modified 8085-type components (and 68000-style components were much more scarce).

Table 1 Development History of Intel Microprocessors

Intel Processor	**Year of Introduction**	**Initial Clock Speed**	**Number of Transistors**	**Circuit Line Width**
4004	1971	108 kHz	2300	10 micron
8008	1972	500-800 KHz	3500	10 micron
8080	1974	2 MHz	4500	6 micron
8086	1978	5 MHz	29000	3 micron
8088	1979	5 MHz	29000	3 micron
Intel286™	1982	6 MHz	134,000	1.5 micron
Intel386™	1985	16 MHz	275,000	1.5 micron
Intel486™	1989	25 MHz	1.2 Million	1 Micron
Pentium™	1993	66 MHz	3.1 Million	0.8 Micron
Pentium™ Pro	1995	200 MHz	5.5 Million	0.35 Micron
Pentium™ II	1997	300 MHz	7.5 Million	0.25 Micron

Celeron™	1998	266 MHz	7.5 Million	0.25 Micron
Pentium™ III	1999	500 MHz	9.5 Million	0.25 Micron
Pentium™ IV	2000	1.5MHz	42 Million	0.18 Micron
Itanium™	2001	800 MHz	25 Million	0.18 Micron
Intel® Xeon™	2001	1.7 GHz	42 million	0.18 micron
Itanium™ 2	2002	1 GHz	220 million	0.18 micron
Pentium™ M	2005	1.5 GHz	140 Million	90 nm

The development history of Intel family of processors is shown in Table 1. The Very Large Scale Integration (VLSI) technology has been the main driving force behind the development.

8.2 A Typical Processor

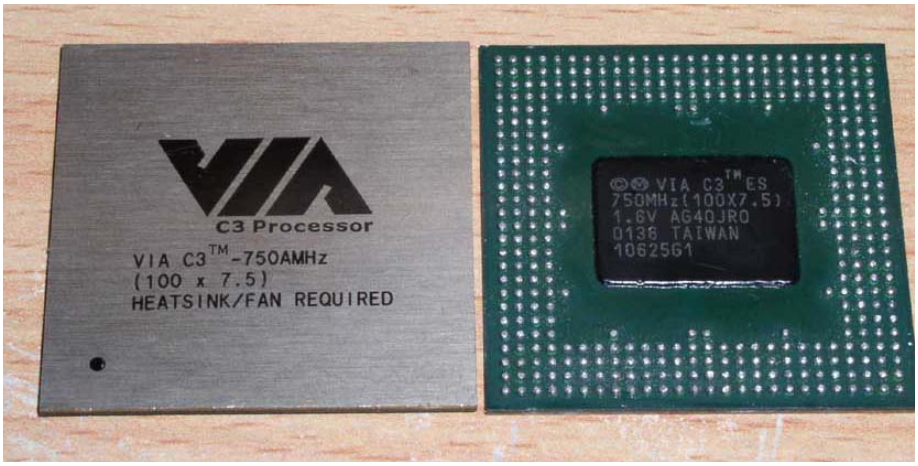


Fig. 8.2 The photograph

The photograph and architecture of a modern general purpose processor from VIA (C3) (please refer lesson on Embedded components 2) is shown in Fig2 and Fig. 8.3 respectively.

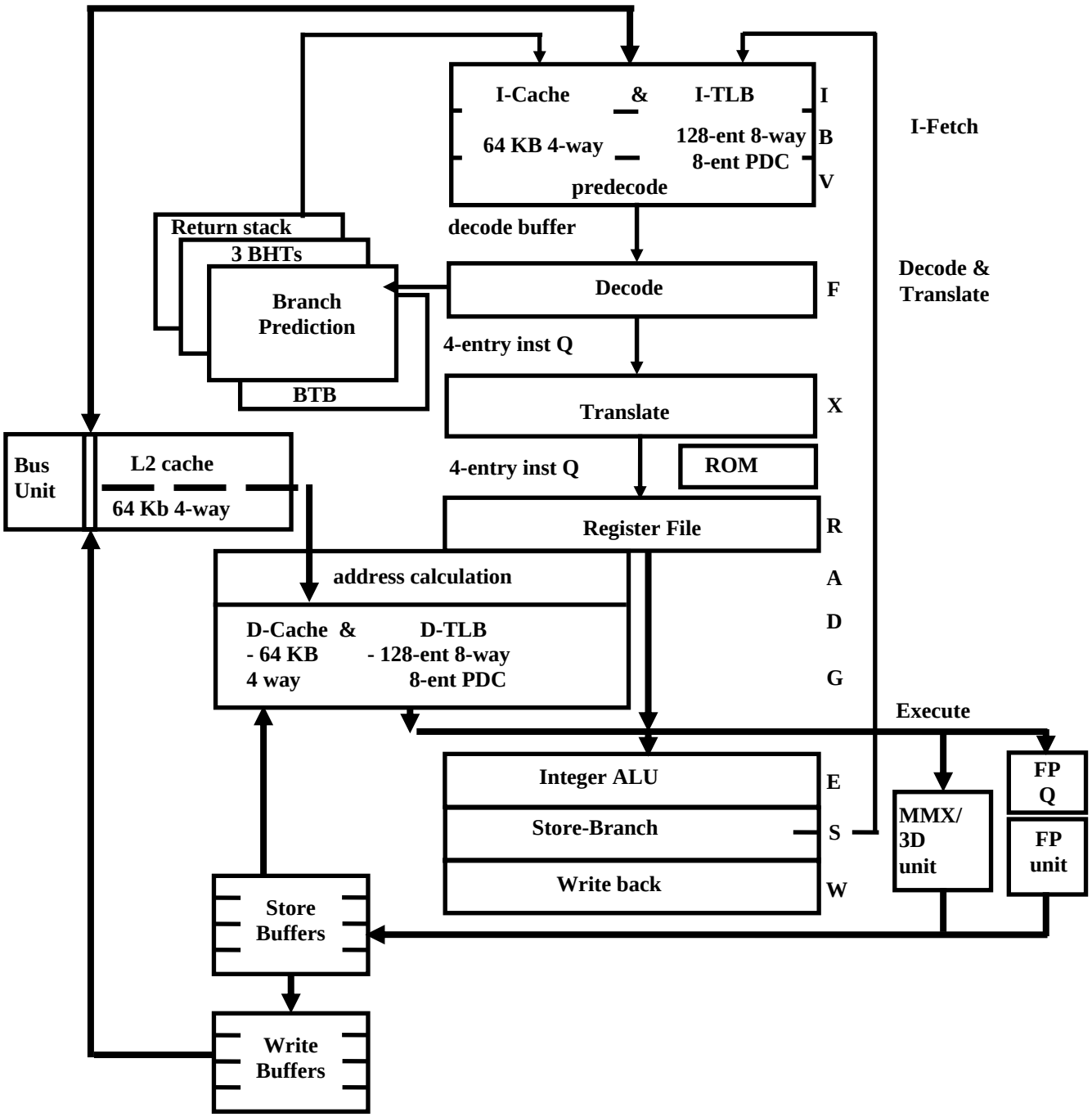


Fig. 8.3 The architecture

Specification

Name: **VIA C3^{TM} in EBGA:** VIA C3 is the name of the company and EBGA for Enhanced Ball Grid Array, clock speed is 1 GHz

Ball Grid Array. (Sometimes abbreviated BG.) A ball grid array is a type of microchip connection methodology. Ball grid array chips typically use a group of solder dots, or balls,

arranged in concentric rectangles to connect to a circuit board. BGA chips are often used in mobile applications where Pin Grid Array (PGA) chips would take up too much space due to the length of the pins used to connect the chips to the circuit board.

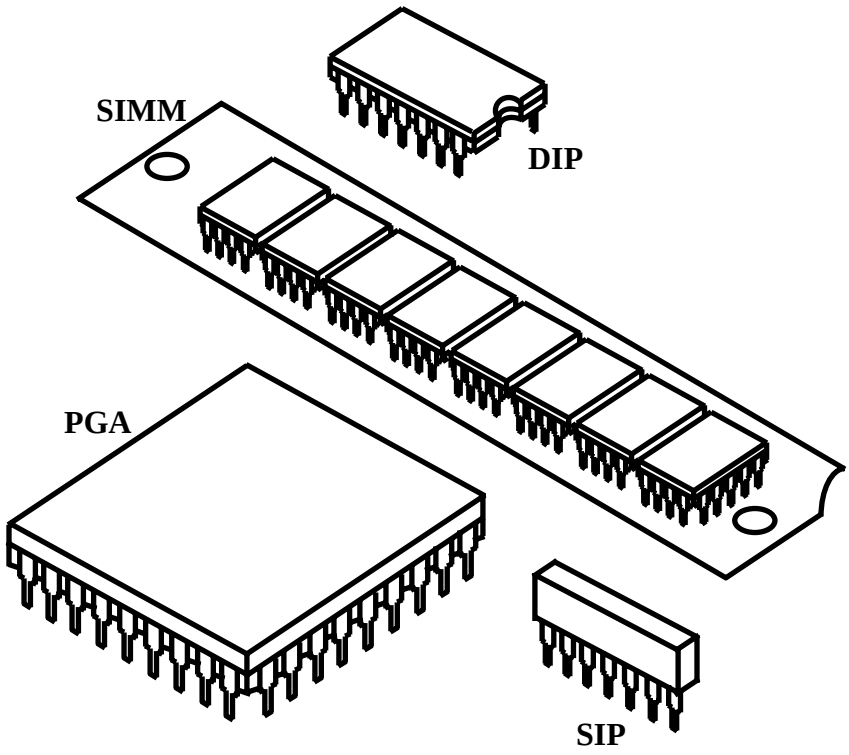


Fig. 8.4 Pin Grid Array (PGAA)

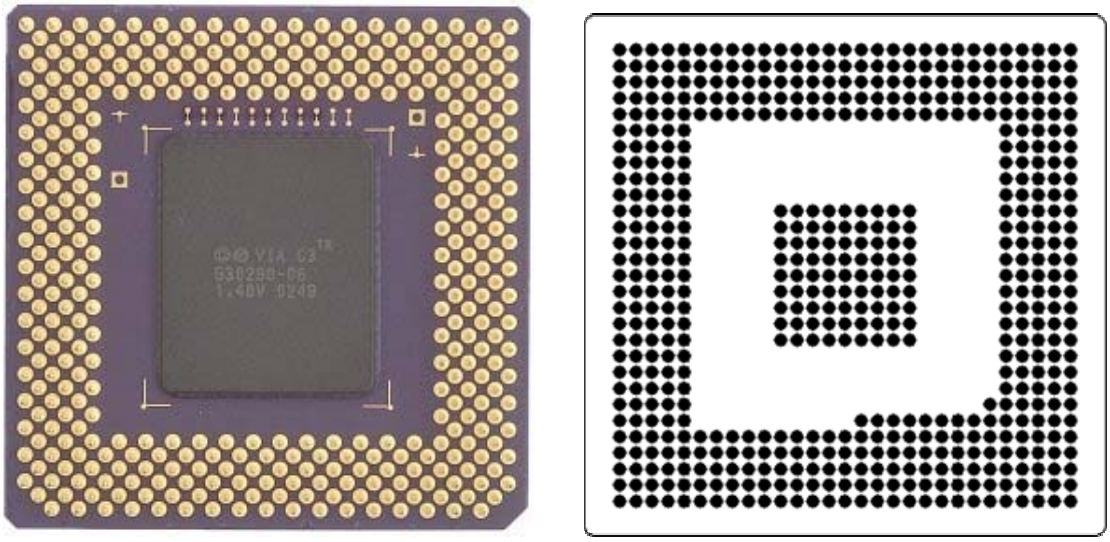


Fig. 8.5 Ball Grid Array

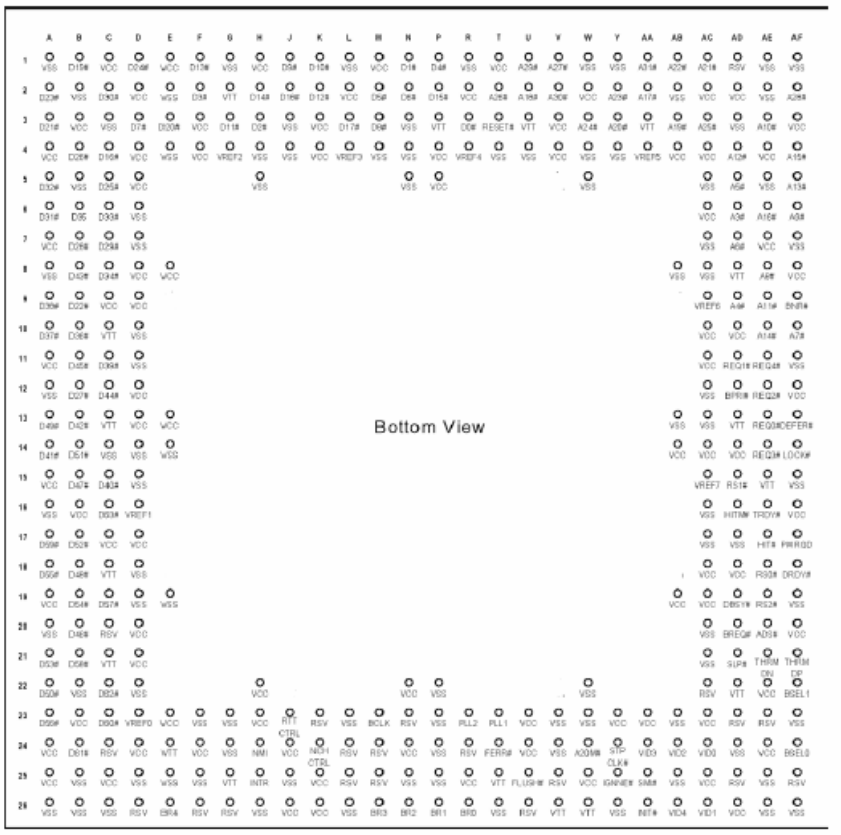


Fig. 8.6 The Bottom View of the Processor

The Architecture

The processor has a 12-stage integer pipe lined structure:

Pipe Line: This is a very important characteristic of a modern general purpose processor. A program is a set of instructions stored in memory. During execution a processor has to fetch these instructions from the memory, decode it and execute them. This process takes few clock cycles. To increase the speed of such processes the processor divide itself into different units. While one unit gets the instructions from the memory, another unit decodes them and some other unit executes them. This is called pipelining. This can be termed as segmenting a functional unit such that it can accept new operands every cycle while the total execution of the instruction may take many cycles. The pipeline construction works like a conveyor belt accepting units until the pipeline is filled and than producing results every cycle. The above processors has got such a pipeline divided into 12–stages

There are four major functional groups: *I-fetch, decode and translate, execution, and data cache.*

- The *I-fetch* components deliver instruction bytes from the large I-cache or the external bus.
- The *decode and translate* components convert these instruction bytes into internal execution forms. If there is any branching operation in the program it is identified here and the processor starts getting new instructions from a different location.
- The *execution* components issue, execute, and retire internal instructions

- The *data cache* components manage the efficient loading and storing of execution data to and from the caches, bus, and internal components

Instruction Fetch Unit

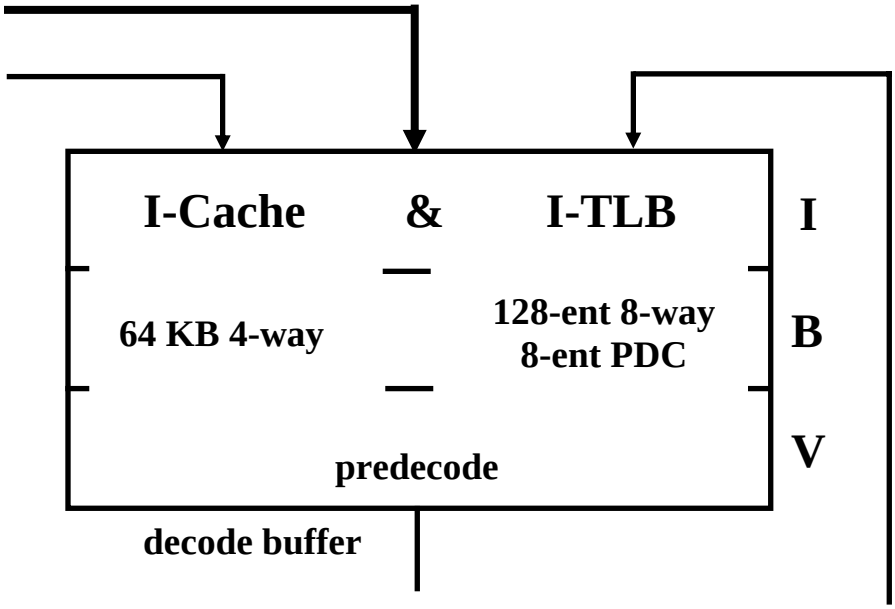


Fig. 8.7

First three pipeline stages (I, B, V) deliver aligned instruction data from the I-cache (Instruction Cache) or external bus into the instruction decode buffers. The primary I-cache contains 64 KB organized as four-way set associative with 32-byte lines. The associated large I-TLB(Instruction Translation Look-aside Buffer) contains 128 entries organized as 8-way set associative.

TLB: **t**ranslation **l**ook-aside **b**uffer
a table in the processor's memory that contains information about the pages in memory the processor has accessed recently. The table cross-references a program's virtual addresses with the corresponding absolute addresses in physical memory that the program has most recently used. The TLB enables faster computing because it allows the address processing to take place independent of the normal address-translation pipeline.
The instruction data is predecoded as it comes out of the cache; this predecode is overlapped with other required operations and, thus, effectively takes no time. The fetched instruction data is placed sequentially into multiple buffers. Starting with a branch, the first branch-target byte is left adjusted into the instruction decode buffer.

Instruction Decode Unit

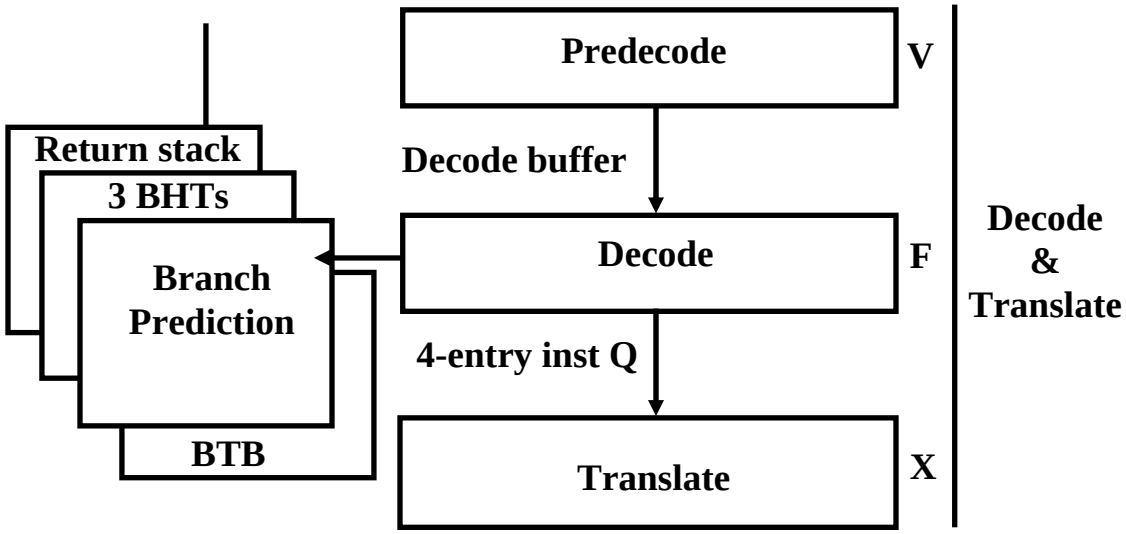


Fig. 8.8

Instruction bytes are decoded and translated into the internal format by two pipeline stages (F,X). The F stage decodes and "formats" an instruction into an intermediate format. The internal-format instructions are placed into a five-deep FIFO(First-In-First-Out) queue: the *FIQ*. The X-stage "translates" an intermediate-form instruction from the FIQ into the internal microinstruction format. Instruction fetch, decode, and translation are made asynchronous from execution via a five-entry FIFO queue (the *XIQ*) between the translator and the execution unit.

Branch Prediction

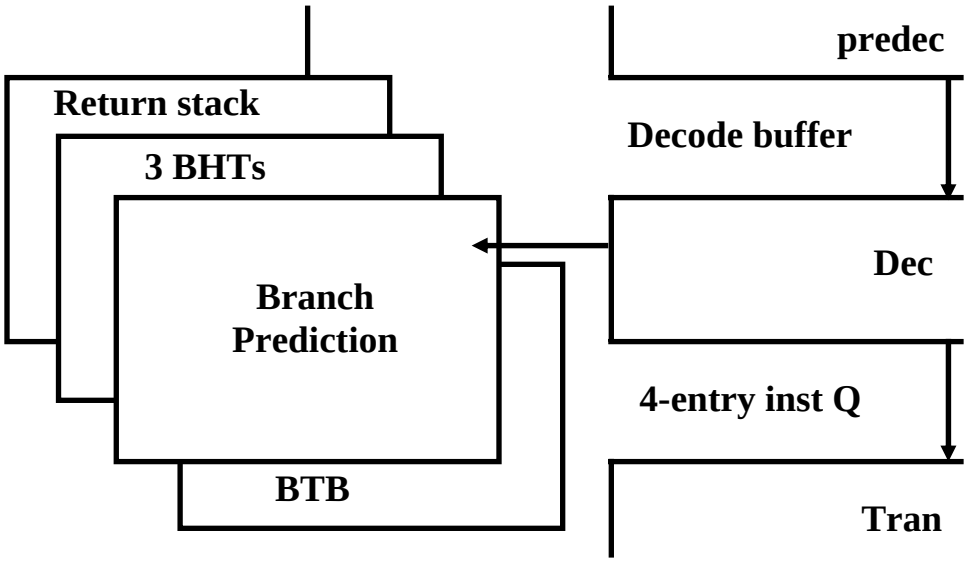


Fig. 8.9

BHT Branch History Table and BTB Branch Target Buffer

The programs often invoke subroutines which are stored at a different location in the memory. In general the instruction fetch mechanism fetches instructions beforehand and keeps them in the cache memory at different stages and sends them for decoding. In case of a branch all such instructions need to be abandoned and new set of instruction codes from the corresponding subroutine is to be loaded. Prediction of branch earlier in the pipeline can save time in flushing out the current instructions and getting new instructions. Branch prediction is a technique that attempts to infer the proper next instruction address, knowing only the current one. Typically it uses a Branch Target Buffer (BTB), a small, associative memory that watches the instruction cache index and tries to predict which index should be accessed next, based on branch history which stored in another set of buffers known as Branch History Table (BHT). This is carried out in the F stage.

Integer Unit

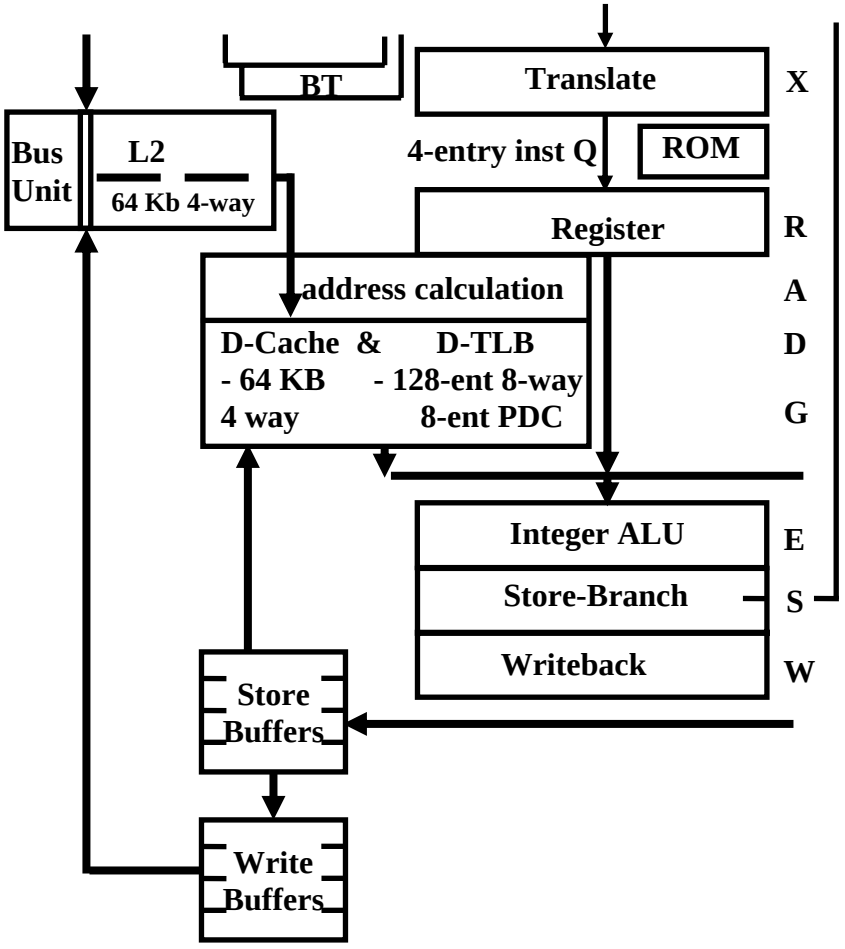


Fig. 8.10

Decode stage (R): Micro-instructions are decoded, integer register files are accessed and resource dependencies are evaluated.

Addressing stage (A): Memory addresses are calculated and sent to the D-cache (Data Cache).

Cache Access stages (D, G): The D-cache and D-TLB (Data Translation Look aside Buffer) are accessed and aligned load data returned at the end of the G-stage.

Execute stage (E): Integer ALU operations are performed. All basic ALU functions take one clock except multiply and divide.

Store stage (S): Integer store data is grabbed in this stage and placed in a store buffer.

Write-back stage (W): The results of operations are committed to the register file.

Data-Cache and Data Path

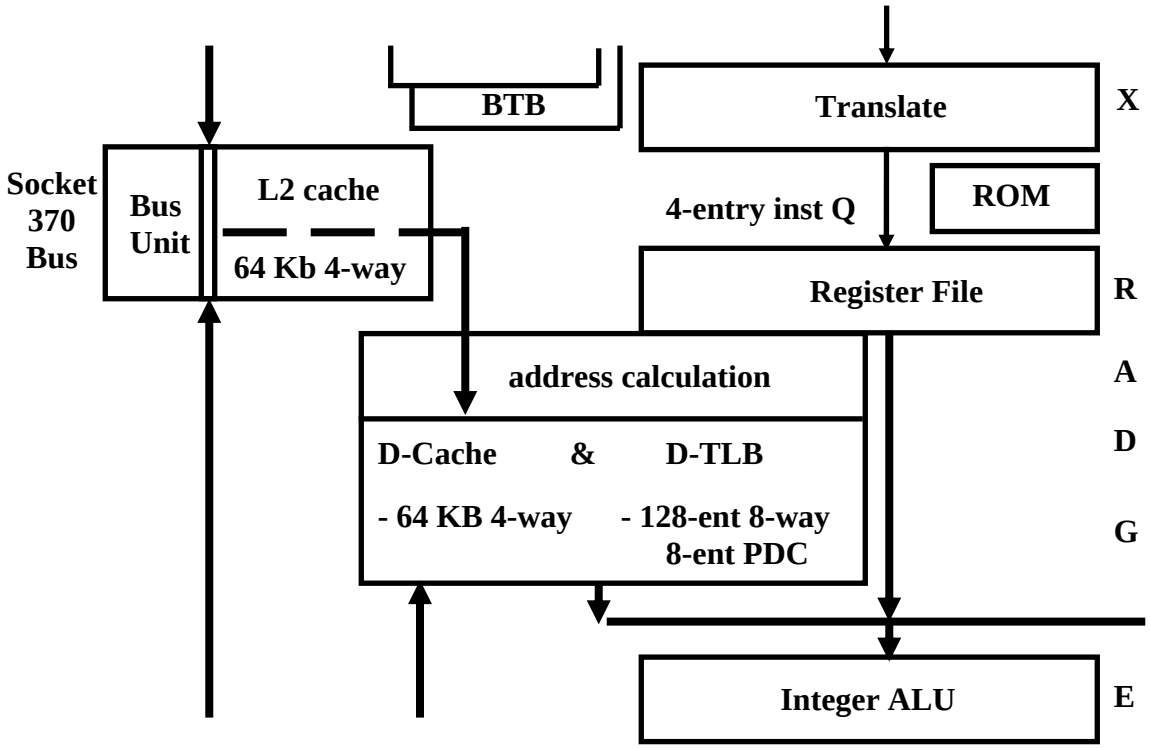


Fig. 8.11

The D-cache contains 64 KB organized as four-way set associative with 32-byte lines. The associated large D-TLB contains 128 entries organized as 8-way set associative. The cache, TLB, and page directory cache all use a pseudo-LRU (Least Recently Used) replacement algorithm

The L2-Cache Memory

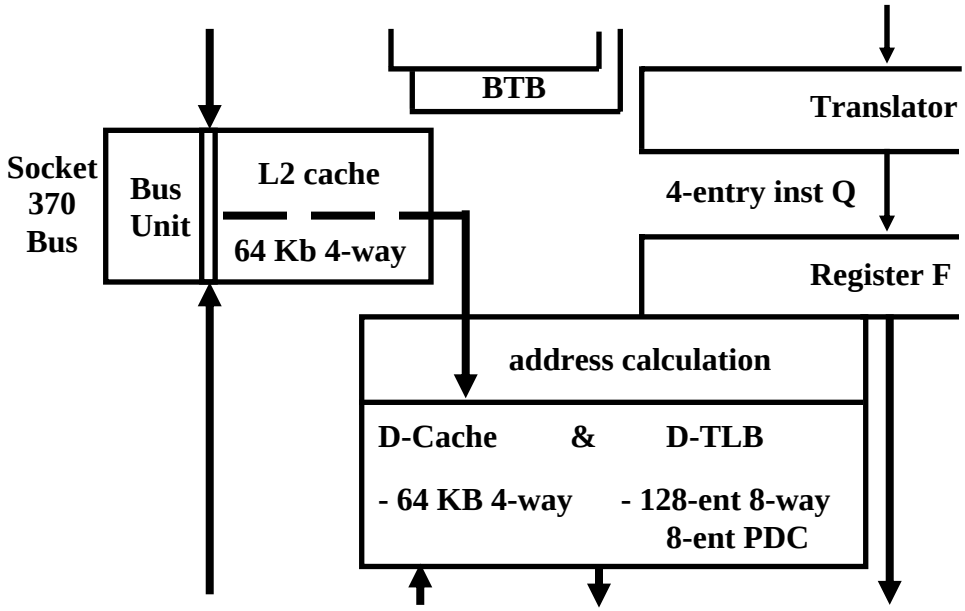


Fig. 8.12

The L2 cache at any point in time are not contained in the two 64-KB L1 caches. As lines are displaced from the L1 caches (due to bringing in new lines from memory), the displaced lines are placed in the L2 cache. Thus, a future L1-cache miss on this displaced line can be satisfied by returning the line from the L2 cache instead of having to access the external memory.

FP, MMX and 3D

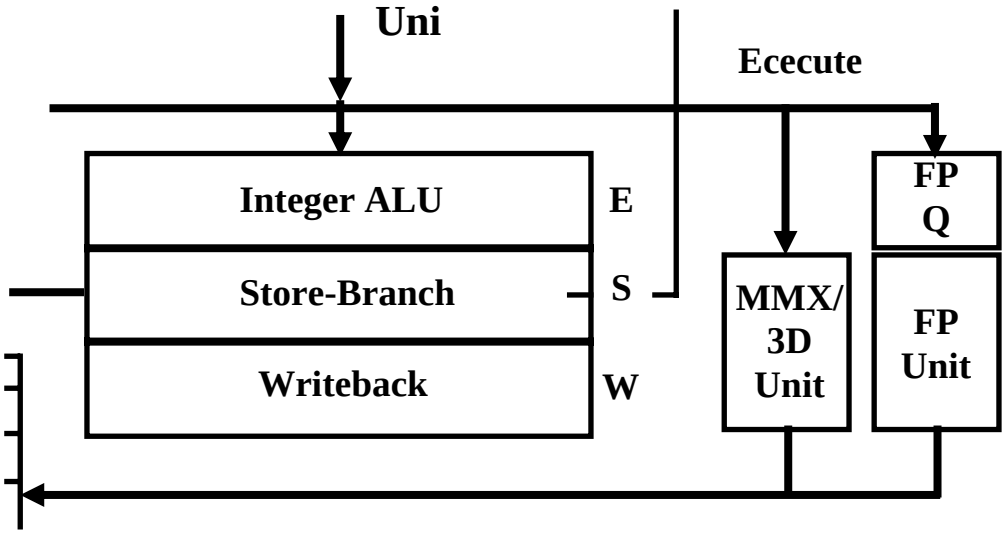


Fig. 8.13

FP; Floating Point Processing Unit
MMX: Multimedia Extension or Matrix Math Extension Unit

3D: Special set of instructions for 3D graphics capabilities

In addition to the integer execution unit, there is a separate 80-bit floating-point execution unit that can execute floating-point instructions in parallel with integer instructions. Floating-point instructions proceed through the integer R, A, D, and G stages. Floating-point instructions are passed from the integer pipeline to the FP-unit through a FIFO queue. This queue, which runs at the processor clock speed, decouples the slower running FP unit from the integer pipeline so that the integer pipeline can continue to process instructions overlapped with FP instructions. Basic arithmetic floating-point instructions (add, multiply, divide, square root, compare, etc.) are represented by a single internal floating-point instruction. Certain little-used and complex floating point instructions (sin, tan, etc.), however, are implemented in microcode and are represented by a long stream of instructions coming from the ROM. These instructions "tie up" the integer instruction pipeline such that integer execution cannot proceed until they complete.

This processor contains a separate execution unit for the MMX-compatible instructions. MMX instructions proceed through the integer R, A, D, and G stages. One MMX instruction can issue into the MMX unit every clock. The MMX multiplier is fully pipelined and can start one non-dependent MMX multiply[-add] instruction (which consists of up to four separate multiplies) every clock. Other MMX instructions execute in one clock. Multiplies followed by a dependent MMX instruction require two clocks. Architecturally, the MMX registers are the same as the floating-point registers. However, there are actually two different register files (one in the FP-unit and one in the MMX units) that are kept synchronized by hardware.

There is a separate execution unit for some specific 3D instructions. These instructions provide assistance for graphics transformations via new SIMD(Single Instruction Multiple Data) single-precision floating-point capabilities. These instruction-codes proceed through the integer R, A, D, and G stages. One 3D instruction can issue into the 3D unit every clock. The 3D unit has two single-precision floating-point multipliers and two single-precision floating-point adders. Other functions such as conversions, reciprocal, and reciprocal square root are provided. The multiplier and adder are fully pipelined and can start any non-dependent 3D instructions every clock.

8.3 Conclusion

This lesson discussed about the architecture of a typical modern general purpose processor(VIA C3) which similar to the x86 family of microprocessors in the Intel family. In fact this processor uses the same x86 instruction set as used by the Intel processor. It is a pipelined architecture. The General Purpose Processor Architecture has the following characteristics

- Multiple Stages of Pipeline
- More than one Level of Cache Memory
- Branch Prediction Mechanism at the early stage of Pipe Line
- Separate and Independent Processing Units (Integer Floating Point, MMX, 3D etc)
- Because of the uncertainties associated with Branching the overall instruction execution time is not fixed (therefore it is not suitable for some of the real time applications which need accurate execution speed)
- It handles a very complex instruction set
- The over all power consumption because of the complexity of the processor is higher

In the next lesson we shall discuss the signals associated with such a processor.

8.4 Questions and Answers

Q1. Draw the architecture of a similar processor (say P4) from Intel Family and study the various units.

Q2. What is meant by the superscalar architecture in Intel family of processors? How is it different/similar to pipelined architecture?

Q3. What kind of instructions do you expect for MMX units? Are they SIMD instructions?

Q4. How do you evaluate sin(x) by hardware?

Q5. What is the method to determine the execution time for a particular instruction for such processors?

Q6. Enlist some instructions for the above processor.

Q7. What is power consumption of this processor? How do you specify them?

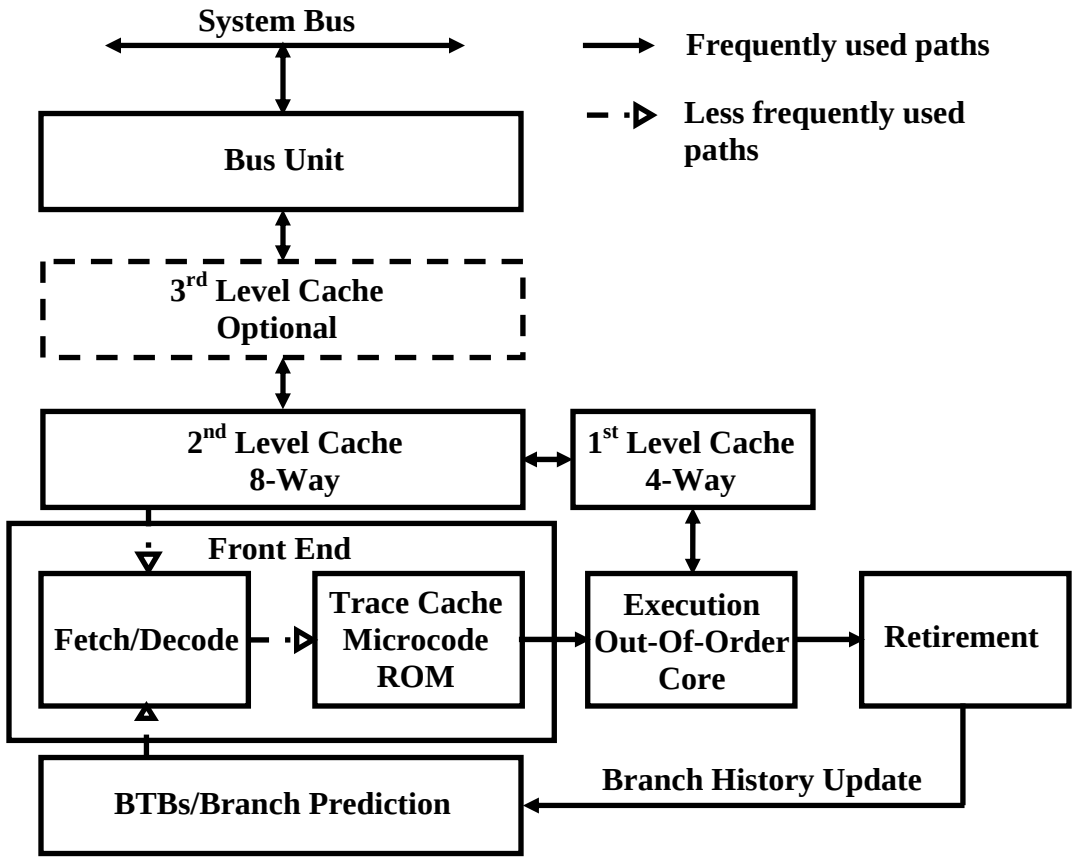
Q8. Give the various logic level voltages for the VIA C3 processor.

Q9. How do you number the pins in an EBGA chip?

Q10. What is the advantages of EBGA over PGA?

Answers

Q1. Intel P4 Net-Burst architecture



Q.2 Superscalar architecture refers to the use of multiple execution units, to allow the processing of more than one instruction at a time. This can be thought of as a form of "internal multiprocessing", since there really are multiple parallel processors inside the CPU. Most modern processors are superscalar; some have more parallel execution units than others. It can be said to consist of multiple pipelines.

Q3. Some MMX instructions from x86 family
MOVQ Move quadword
PUNPCKHWD Unpack high-order words
PADDUSW Add packed unsigned word integers with unsigned saturation
They also can be SIMD instructions.

Q4. (a) Look Up Table
(b) Taylor Series
(c) From the complex exponential

Q5. This is done by averaging the instruction execution in various programming models which includes latency and overhead. This is a statistical measure.

Q6. All x86 family instructions will work.

Q7. around 7.5 watts

Q8.

Parameter	**Min**	**Max**	**Units**	**Notes**
V_{IL} – Input Low Voltage	-0.58	0.700	V	
$V_{IH1.5}$ – Input High Voltage	V_{REF} + 0.2	V_{TT}	V	(2)
$V_{IH2.5}$ – Input High Voltage	2.0	3.18	V	(3)
V_{OL} – Low Level Output Voltage		0.40	V	@I_{OL}
V_{OH} – High Level Output Voltage		V_{CMOS}	V	(1)
I_{OL} – Low Level Output Current	9		mA	@V_{CL}
I_{LI} – Input Leakage Current		±100	μA	
I_{LO} – Output Leakage Current		±100	μA	

Q9. Refer Text

Q10. Refer Text