Alpha chip.

Alpha was designed by Digital as a 64-bit (load and store) architecture.

- All registers are 64 bits in length.
- All operations (data manipulation) are performed between registers.
- External access are only memory operations such as loads or stores

Instructions are coded as 32 bits (longword / 4 bytes) sequences.
Is it confusing?
Example:

addq Register1, Register2, Register3

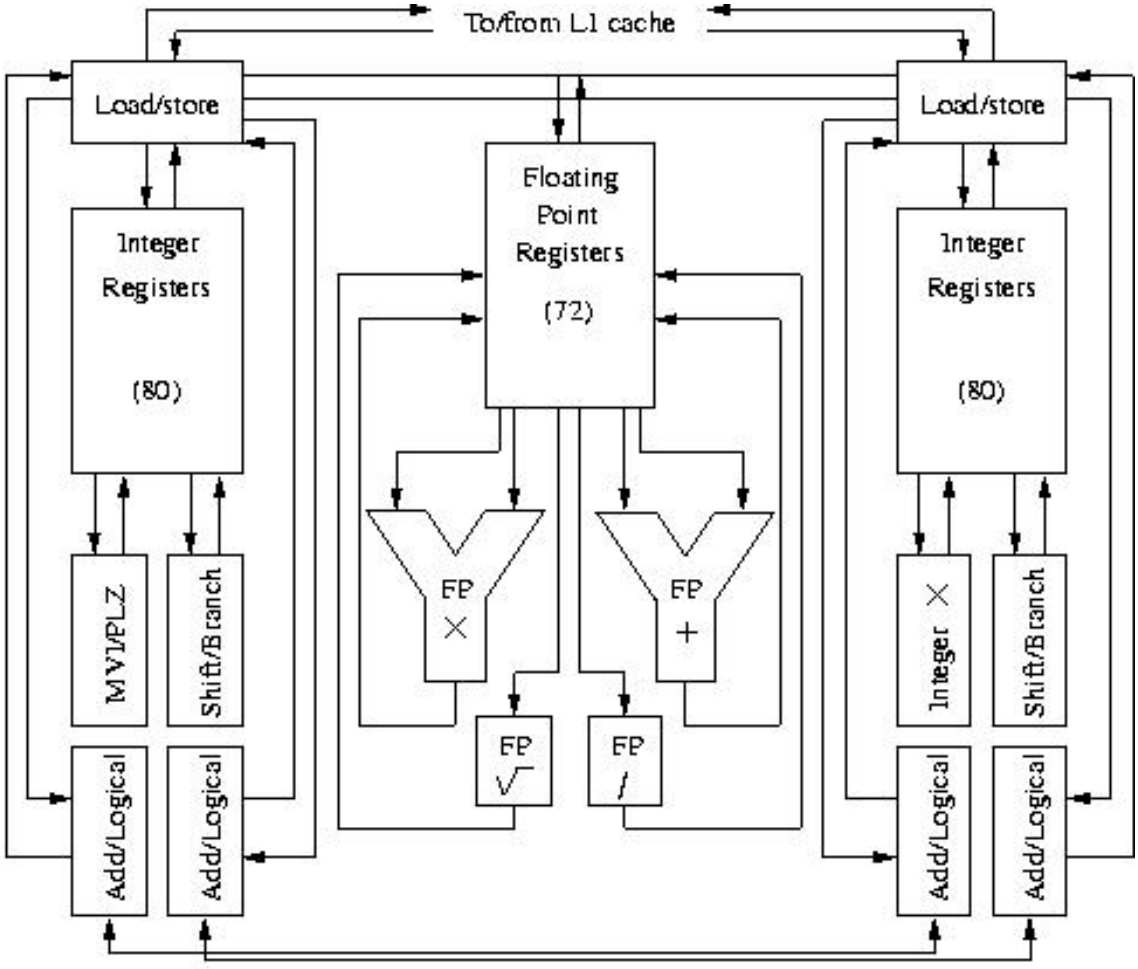
Stands for the addition between what is in register1 and register2 with the results being contained by register3.
Addq stands for addition between quadwords (e.g 64 bits long number).
Therefore each register contains a quadword usually displayed in its hexadecimal form:

Register1 contents=0x123456789abcdef0 just an example

The instruction has to be coded to be interpreted and executed by the processor.
It is coded as 32-bit long value which will contain identification code (usually called opcode) of the operation performed and information on which registers are involved.

The above instruction is coded as *0x40220403*

Functionnal diagram of an alpha EV-7 processor

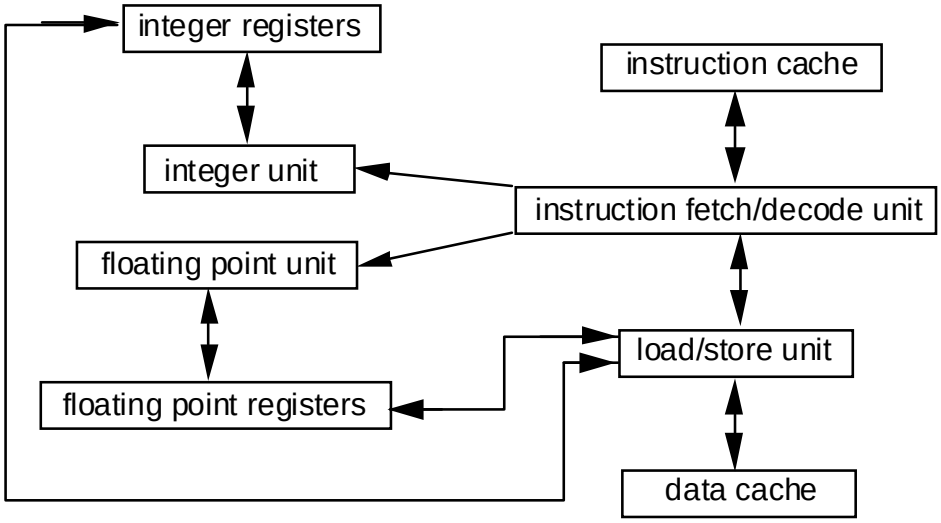


(a)

Features:

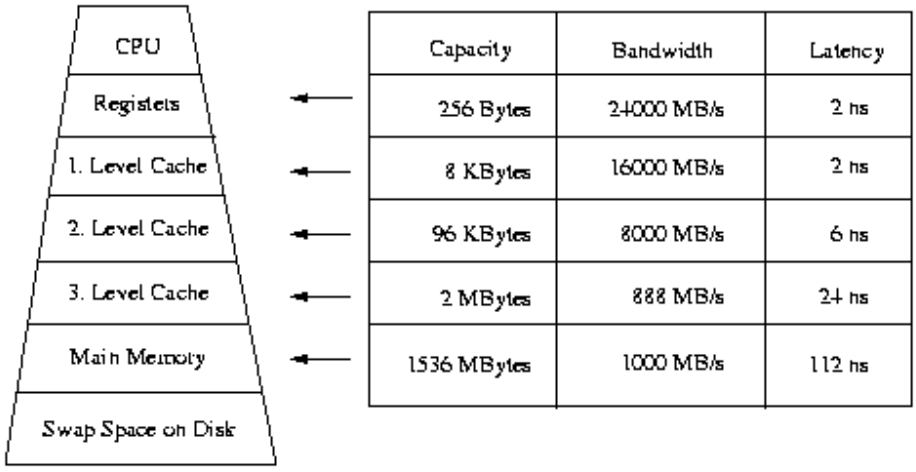
- *Two* duplicate integer register units.
- Four integer Add/Logical units (exchange values in one cycle if required).
- Two load/store units draw on a 64 KB instruction and a 64 KB data cache.
- 80 integer and 72 floating-point registers.
- 4 dual channels (North, East, South, West) from the chip to interconnect it with neighbouring chips at a bandwidth of 1.6 GB/s per single channel.
- direct I/O dual link from the chip has a bandwidth of 1.6 GB/s
- The chip is expected to ship first at a clock frequency of 1-1.2 GHz.

Simplified Functional block diagram of the 21164 Alpha chip



Features of the 21164 Alpha chip

- 64 bit RISC architecture
- 32 integer registers
- 32 entry, 64 bit floating-point register file
- 8 KB, direct-mapped, L1 instruction cache (onchip)
- 8 KB, direct-mapped, write through L1 data cache (onchip)
- 96 KB, L2 data and instruction cache (onchip)



Memory hierarchy

ALU Units:
Arithmetic/Logic Unit where arithmetic and logical operations of the CPU are performed. Different units are in charge of integer, floating points and address operations.

Integer unit: This unit carries out arithmetic (addition, subtraction) and logic (shift left or right) operations on integers.

Integer registers: Integer registers store the integers used during operations

Instruction cache: The instruction cache stores the instructions to be executed by the CPU.

Data cache: Data cache holds the values of the variables in the programs.

- Local memory, which may be addressed several times faster than the primary memory. Used to speed up time-consuming tasks such as loading/unloading instructions/data from memory.
- The bigger, the better (16 KB -> 112 KB so far)

Load/store unit: This unit is responsible for controlling the data transfer between the data cache and the register.

Instruction fetch/decode unit (control unit):

- This unit fetches the instructions from the instruction cache.
- The instructions are interpreted; and, according to the instructions, the unit issues signals to the other components in the CPU to control their operations.
- Program counter (PC) in this unit. The PC stores the address of the next instruction to be executed. The unit fetches the next instruction according to this address. After an instruction is fetched, the value of the PC is updated to point to the next instruction in the memory.

Class-like definition of the Alpha Architecture

Alpha was designed by Digital as a 64-bit (load and store) architecture.

- All registers are 64 bits in length.
- 32 integer registers and 32 floating point registers.
- All instructions are 32 bits in length (longword / 4 bytes).
- All operations are performed between 64-bit registers.
- Memory operations are either loads or stores.

There is a program counter (PC) register, which contains the memory address of the next instruction, 32 integer registers to contain integer data, and 32 floating point registers, to contain floating point data.

- CPU is roughly corresponding to the following data structure:

```
class CPU
    {
    Quadword pc;
    Quadword[] intReg = new Quadword[ 32 ];
    Quadword[] floatReg = new Quadword[ 32 ];
    // and some special registers
    }
```

intReg[31] and floatReg[31] always return zero if read, and writing to them has no effect.

- A way to store 0 somewhere (will see later)

Memory corresponds to an array of bytes.

```
    byte[] memory = new byte[ ... ];
```

A computer executes instructions. The instructions are stored in computer memory. The program counter register contains the address of the next instruction. The CPU loops, obtaining an instruction, incrementing the program counter, decoding and executing the instruction.

```
while ( true )
    {
    Longword instruction = getLongwordAt( pc );
    pc = pc + 4;
    decode the instruction;
    obtain the operands of the instruction;
    perform the operation of the instruction;
    save the result;
    }
```

On the Alpha, each instruction is stored in a longword (32 bits or 4 bytes). Instructions must be aligned (stored at an address divisible by 4).
Normally, instructions at successive addresses are executed in sequence.

- The program counter is incremented by the size of an instruction after loading the instruction into the CPU

Some instructions (*branch instructions*) can modify the PC.

- The way to deviate from a straight line path of execution to create loops, if statements, etc.

The Program Counter (PC) is a special register that addresses the instruction stream. As each instruction is decoded, the PC is advanced to the next sequential instruction. This is referred to as the *updated P*C. Any instruction that uses the value of the PC will use the updated PC. This quantity is a long-word-aligned byte address. The PC is an implied operand on conditional branch and subroutine jump instructions. The PC is not accessible as an integer register.