Graduate Student Assignment

Name: Avineet Kumar Singh

Analysis of 3 Research Literature

Research Paper 1: The Tera Computer System

By Robert Alverson, David Callahan, Daniel Cummings, Brian Koblenz, Allan Porterfield, Burton Smith

The Tera Computer System was built to be suitable for very high-speed computation, which is equivalent to having a short clock period. It was made to be scalable to many processors, applicable to different types of problems, even to those that do not vectorize well, and was made to have easy compiler implementation.

Implementation of 3 major objectives in 'Tera Computer system'.

1) To build a system which is suitable for very high-speed implementation.

Interconnection Network

Its interconnection network features a uniform distribution of resources. That is, every processor has equal access to every memory location. This makes it easy to program because concerns for the memory layout are eliminated. On the other hand, if p is the number of processors, the number of network nodes grows as p3/2 rather than plog(p) which results when using multistage networks. That is, the architecture has some additional overhead per processor. Bandwidth of the network is very high helping to implement instructions at high speed.

Memory

Data addresses are randomized in the processors.

The randomization is excellent for avoiding memory bank hotspots and network congestion but makes it difficult to exploit memory locality using nearby memory units. In the Tera system, the randomization is combined with another notion called distribution helping it to exploit memory locality and increasing the speed.

A processor fetches instruction through a special path to a neighboring I/O cache unit. This avoids net- work traffic and network latency.

Processors

Each processor in a Tera computer can execute multiple instruction streams simultaneously.

The lookahead allows streams to issue multiple instructions in parallel, thereby reducing the number of streams needed to achieve peak performance.

2) To build a system which solves wide range of problems.

Horizontal Instructions

The utilization of the instruction interpretation resources has always been constrained by the difficulty of issuing more than one instruction per tick.

Vector instructions sidestep this difficulty in part but are not able to handle frequent conditional branches or heterogeneous scalar operations well.

In a horizontal instruction, several operations are specified together.

Tera instructions are mildly horizontal.

Vectorizable loops can be processed at nominal vector rates (one flop per tick) using only horizontal instructions with these three kinds of operations.

Matrix-vector multiplication attains nearly two flops per tick via the same technique used for its efficient vectorization.

3) To build a system which simplifies compiler implementation.

Exceptions

The decision to preserve operand values for possible use by the trap handler is made by the compiler.

Trap handling is extremely lightweight and independent of the operating system. Trap handlers can be changed by the user to achieve specific trap capabilities and priorities without loss of efficiency. Hence simplifying compiler implementation.

Research Paper 2: Design of the B 5000 System

By William Lonergan and Paul King.

The B5000 was designed with the support of high-level languages (ALGOL and COBOL) in mind. The most remarked-upon aspects are its use of a hardware-managed stack for calculation, and the extensive use of descriptors for data access. It included virtual memory -- perhaps the first commercial computer to do so -- as well as support for multiprogramming and multiprocessing. For additional security, code and data were distinguished in memory using a "flag bit", and the hardware would not execute data or alter code without first having explicitly changed the flag.

High Level Language Support

The Burroughs B5000 was designed from the start to support high level languages (HLLs), specifically ALGOL and COBOL. The initial features supporting this were the hardware-managed stack, the descriptor, and segmentation. A redesign of this system, the B6000, extended the concept by moving the tag bits to their own field, external to the 48-bit data word.

Stack Architecture

All calculations performed on B5000 machine take place using the operands at the top of the stack. To improve performance, the top two stack locations exist as the A and B registers (each of which may be extended sideways by the X and Y registers for double-precision operands). Pushing and popping the stack is done automatically by the hardware, depending upon the specific operations being done. This makes for rapid expression evaluation, and greatly simplifies a compilers task of code generation, a fact recognized in other designs. Since the operands are always implied (the stack top), instructions do not need an operand field, reducing code size.

The Instruction Set

Instruction set by category:

- Arithmetic Instructions
- String Instructions
- Multitasking Support
- Data Structure Support
- User Definable Instructions

Research Paper 3: Architecture of the IBM System/360

By G.M. Amdahl, G.A. Blaauw, F.P. Brooks,Jr.

The IBM System /360 ushered in an era of computer compatibility—for the first time, allowing machines across a product line to work with each other. In fact, it marked a turning point in the emerging field of information science and the understanding of complex systems.

The IBM System/360 architecture is the model independent architecture for the entire S/360 line of mainframe computers. It made a significant decision to be binary compatible with their machines and make all of the machines work off a common instruction set.

Memory

Memory (storage) in System/360 is addressed in terms of 8-bit bytes.

Addressing

System/360 uses truncated addressing.

The S/360 architecture defines following formats

- characters
- integers
- decimal integers
- Floating point

Instruction Formats

Instructions in the S/360 are two, four or six bytes in length, with the opcode in byte 0. Instructions have one of the following formats:

- RR
- RS
- RX
- SI
- SS

Input/Output System

I/O is carried out by a conceptually separate processor called a channel. Channels have their own instruction set, and access memory independently of the program running on the CPU.

There are three types of channels on the S/360:

- byte multiplexer channel
- selector channel
- block multiplexer channel

Comparison of above 3 architectures

One significant introduction from the IBM System/360 was byte address-ability, whereas the B5000 was word addressed. Byte address-ability allowed a bigger address size than word size, and the ability to use characters located at any location.

The B5000 used a stack-based register style. This means that there are no registers available to the assembly programmer, and anything that needs to get referenced must be pushed on to the stack. Another important side-effect of the stack-based architecture was that the B5000 descriptor system used a base and limit. This, along with the separation between instructions and data, removed the possibility for stack overflow attacks.

Tera machine separates operations into memory operations, and control operations. The unusual custom designed processor can issue three operations per instruction either one from each category or two arithmetic operations and one memory operations.

Tera Architecture has strong support for implementing non-numeric languages like Lisp and Prolog and highly applicative languages like Sisal and Id, whereas Burroughs B5000 was designed from the start to support high level languages (HLLs), specifically ALGOL and COBOL.

The IBM System/360 and the B5000 differed in many fundamental ways, below table shows just some of the differences:

	IBM System/360	**Burroughs B5000**	**Tera Computer System**
Address size	24 bits	Program Reference Table with 1024 entries	48 bit

Character size	4 bit for binary coded decimal	6 bit (8 characters for 48 bit word)	64-bit words
FP size	32/64 bit	48 bit	32/64 bit
Instruction size	Variable: 16/32/64 bit	12 bit	32/64 bit
Integer size	32 bit	48 bit	8/16/32/64 bit
Register style	General purpose registers	Stack based	Stream based/stored in bank (organized into multiple banks with each bank containing all of the registers)

References

Research Paper 1: The Tera Computer System

Research Paper 2: Design of the B 5000 System

Research Paper 3: Architecture of the IBM System/360

https://www.smecc.org/

https://msujaws.wordpress.com/

Wikipedia

Note:

Some lines are directly taken from the research paper provided as I could not explain it in a better way as it was given.