Homework 1

CIS-655 ADVANCED COMPUTER ARCHITECTURE

PROF. MOHAMMED ABDALLAH

10/15/2021

Anthony Redamonti
SYRACUSE UNIVERSITY

Question 1:

Article 1: Architecture of the IBM System/360

The article written by G.M. Amdahl, G.A. Blaauw, and F.P. Brooks, Jr. details the architecture of the IBM System/360. The focus of the paper is to discuss the need for compatible CPU's that can provide real-time processing. It also discusses the issues that were faced and overcome throughout the design process.

The design objectives for the system were as follows:

The system would need to be compatible for I/O devices of various size and data throughput. Special care must be made in the design of the compiler, I/O management, and memory management as these functions would be executed often. Therefore, they must be as efficient as possible. Machine systems must be able to manage themselves without user input in both real-time and non-real-time applications. The system must allow redundant systems to be incorporated to handle I/O, memory, or CPU module failure. Large storage capacities must be required (more than 32,000 words). The system must support the use of 36-bit floating point integers. The CPU must check for hardware malfunction in all systems with the aid of built-in hardware fault-locating aids to reduce down-times.

Stack vs. The Addressed-Register Technique

One key decision of the machine was the choice to use an addressed-register organization implementation instead of a pushdown accumulator (stack). Several drawbacks of stack implementation are that the depth of the stack has a limit, the recursive subroutines require additional independent stacks for addressing purposes, and fitting variable-length fields into a fixed-width stack is awkward. Another issue is the range of storage addresses. Larger models would require storage capacities in the millions of characters. The addressed-register organization was able to reduce address appearances in the instruction stream by a factor approaching 2. All addresses had to be indexable; the technique proved effective in past designs. A new development was the use of a base register, which provided a more abundant address size. The decision also allowed for gains in instruction density. The decision was made to make the base-register technique the main means of register addressing and to use only 12 bits for direct addressing. Direct addressing would therefore only be advantageous for very small modules.

The Input/Output System

Small machines use the CPU hardware for I/O functions whereas large machines have hardware independent of the CPU to handle data transfer, such as DMA (Direct Memory Access). The method of I/O communication in the IBM system utilizes *channel instructions*. The channel between a peripheral device and memory is considered an independently operating entity. The CPU specifies the beginning of a channel program and the unit to be used, and the channel instructions specify the storage blocks to be read or written. When the channel program ends, the CPU is interrupted, and the channel is made available again. In smaller models the flow of data between I/O devices and memory requires the use of the CPU, which increases interference with the CPU and lowers the maximum data rate.

Article 2: Design of the B-5000 System

The B-5000 system was designed with the objectives of achieving high-level machine independent languages, efficiency in machine language compilation, debugging high-level languages, and reducing the problem set-up and load time. The design criteria stated that programs should be independent of their location in memory. The addressing of memory should take advantage of contextual addressing schemes to reduce redundancy.

A master control program is used when the system is in the "Control State." All operations in this state are made via interrupts. A special operation is provided, which will return control to the object program which was executing at the time of the interrupt. Common causes of interrupts in the system pertain to I/O functions and invalid indexing operations.

Single-address computers waste run-time by storing and recalling intermediate results in sequence computations. The B-5000 implements a pushdown stack, which eliminated the need for instructions to store or recall intermediate results. The B-5000 processor uses a pair of registers (A and B) to eliminate unnecessary pushdown/pullup stack operations, saving valuable run-time.

As previously stated, one of the goals of the B-5000 was to make programs independent of their actual memory location. By achieving independent addressing, programs would no longer need to be contiguous in memory. They can be broken up into blocks through a process known as segmentation. It is then possible to load a program into core memory at run-time as needed.

Article 3: The Case for the Reduced Instruction Set Computer

The following paper makes a compelling argument for why the next generation of computers should be implemented using a Reduced Instruction Set Computer (RISC) method rather than a Complex Instruction Set Computer (CISC) method.

There are several consequences of implementing the CISC architecture. Due to the advances in semiconductor memory, the difference in speed between the CPU and main memory has been reduced, making CISC less necessary. Another factor to consider is that a special purpose instruction may not be faster than a sequence of simple instructions. For example, a sequence of simple load instructions is faster than one complex load instruction in typical programs. CISC is also very difficult to design into a system due to its complexity, creating longer design times and increased design errors.

There are several advantages to using the RISC architecture. A chip design must be able to fit an entire CPU (SOC – system on chip). A CPU using the complex architecture has less likelihood of being able to fit on a single chip than one implementing a RISC architecture due to its better use of chip area. It is also easier to design into an existing system and is more cost effective. It can also support high-level language computer systems since it can discover and report syntax and execution errors at run-time and hide the internal language implementation from the user. RISC architecture is currently being studied at Berkley University, Bell Labs, and IBM to name a few well-known, respected scientific organizations.

In conclusion, CISC may benefit the execution time of a program but rarely does it reduce the execution time of the entire system.

Article 4: Comments on "The Case for the Reduced Instruction Set Computer"

Douglas W. Clark and William D. Strecker argue that some of the points made in the previous article "The Case for the Reduced Instruction Set Computer" are misleading. They regard the paper as a casual evaluation of RISC vs. CISC and note that to make concrete conclusions of their comparison, an in-depth study of hardware design, microcode, processor construction and performance across a variety of applications is necessary.

Clark and Strecker reprimand the paper for not clearly defining RISC or CISC and state that instruction set complexity cannot be measured by instruction count alone. It should also consider the number of data types and completeness. Also, due to recent improvements in memory size, code density is not as important as it once was. Dense code also offers improvements in cache and paging with more instructions per cache block and per page.

Clark and Strecker argue that replacing rarely executed instructions with a multi-instruction sequence can make execution time much worse for compilers. The purpose of a large instruction set architecture in a CISC system is to keep the operators and data types orthogonal. One advantage CISC has over RISC is that specialized hardware can be easily implemented to improve cost and performance of the system. For example, consider the multiply function on the CISC. To improve its performance, simply add a specialized data path. The multiply function is not available on the RISC system, so it must perform a series of moves, branches, shifts, and adds. To improve its performance, the entire processor would need to be upgraded.

Question 2:

$$Die\ Yield = Wafer\ Yield * \left(1 + \frac{Defects\ per\ unit\ area * Die\ area}{N}\right)^{-N}$$

$$Die\ Yield = 1 * \left(1 + \frac{\left(0.03 * \frac{389mm^2}{100mm^2}\right)}{13.5}\right)^{-13.5}$$

$$Die\ Yield = \left(1 + \frac{0.03 * 3.89}{13.5}\right)^{-13.5}$$

$$Die\ Yield = \left(1 + \frac{0.1167}{13.5}\right)^{-13.5}$$

$$Die\ Yield = (1.0086444)^{-13.5}$$

$$Die\ Yield = 0.890298$$

Question 3:

A cooling door for a rack costs $4000 and dissipates 14kW (into the room; additional cost is required to get it out of the room). How many servers with an Intel Pentium 4 processor, 1 GB 240-pin DRAM, and a single 7200 rpm hard-drive can you cool with one cooling door?

$$Peak\ power\ consumption\ per\ system = 66W + 2.3W + 7.9W = 76.2W$$

$$Amount\ of\ systems\ able\ to\ be\ cooled: \frac{14{,}000W}{76.2W} = 183$$