IBM POWER 7

Julia Masciarelli [1]

Jason Wu [2]

Johnny Halter [3]

CS440 Computer Organization Architecture

Regis University

Professor Pam Smallwood

**Architecture of IBM POWER 7 Processor**

[1][2][3] IBM has been a leader in computing throughout the technological age. As one of the oldest computing companies in the United States, it quickly rose to the top of the industry and has maintained this position with their innovative solutions to the world's problems. IBM has developed countless computer processors during their business, and with each new processor comes better technology and hardware advancements. The IBM POWER7 is one of these processors, which has replaced its predecessors with even better performance but is not the stopping point for IBM. This paper examines the details of the processor from a hardware standpoint, and also discusses the unique features that set POWER7 apart from its competitors. The paper describes the details of the instruction set architecture in the POWER7, and how the different instructions are classified. It also gives three examples of instructions that are supported in the instruction set architecture and how each bit in the instruction is interpreted by the CPU. In the POWER7 architecture, there are pipelined processes, and a full high level overview of the pipeline is described as well as three in depth explanations of different instructions that use sections of the pipeline. Although there are many advantages to the POWER7 processor, there are also disadvantages to using the specific processor. Both sides of this argument are examined in the paper. The paper also summarizes the history of IBM as a company and their reasoning for developing the POWER7.

**History**

**History of IBM**

[1] On June 16th, 1911, IBM opened its doors as the Computing-Tabulating-Recording Company (CTR). The company founder, Thomas J. Watson Sr. brings the motto of "Think" to IBM. "Thought has been the father of every advance since time began" (Madrigal, 2011), states Watson Sr., which sparked innovation at CTR. In 1924, International Business Machines (IBM) replaced CTR. From there, modern IBM began creation. The United States transitions to using Social Security in 1935, and entrusted IBM's punched card machines to keep the records of the tens of millions of Americans using Social Security, and the IBM tabulating machine is integral in war efforts in 1942, which was used to keep track of metrics such as freight train traffic and other important statistics during WW2. During the war, technology innovation was at a boom. In 1944, the first computer at IBM, the Automated Sequence Controlled Calculator or Mark I was developed. The Mark I stood at 51 feet long, weighing 5 tons. To complete an addition operation took 1/3 of a second, and a multiplication operation took 1 second. Mark, I took inputs through a punch card and printed paper outputs through a typewriter. The Mark I assisted the Navy by calculating gun trajectories during WW2. In 1945, WW2 ended and the Nuremberg Trials were held to discuss war crimes punishments. IBM's technology was used to translate the trials. The system would translate predetermined speeches into different languages, such as English, Russian, French, and German, and then play the audio back at the same time it was spoken. Although this system did not allow for real-time translation, it was one of the first translation systems developed. The IBM 305 RAMAC was one of the groundbreaking computers of its time, developed in 1956. It was the first commercial computer that used a hard disk drive for secondary storage. RAMAC stood for Random Access Method of Accounting and Control. In the year 1960, IBM reached 100,000 employees. At this point, computers are still not a household item, so the majority of IBM computers were created for large organizations. During this decade, the American industry was booming in a post-war era, and business thrived. IBM computers were utilized for their enormous memory and computing capabilities. The IBM computing power was undoubted as business leaders turned to their machines both to “manage and produce massive amounts of data” (Madrigal, 2011), securing IBM’s spot as a leader in computing. In 1980, IBM made a deal with Microsoft to run their software on all of their computers. A year after this partnership, the first IBM computer made for “everyday people” was released - the IBM Personal Computer 5150 (Madrigal, 2011). This was a milestone transition in computing because computers were no longer just for the government and big businesses. The Personal Computer 5150 weighed 21 pounds without any disk drives, kilobytes of memory space, and took 410 nanoseconds to complete a main storage cycle (Madrigal, 2011). Soon after IBM, personal computers became one of the company’s most popular devices. In the early 90s, IBM decided that it was time to modify their business strategies. For so long they had been known for their high power business machines, but the new age of computing was developing, and IBM knew it was time to become “a world-class services company” (Madrigal, 2011). IBM turned towards making software and technological products, and became an innovative leader in new computer technology. The world was amazed when IBM Deep Blue beat Garry Kasparaov, the reigning world champion in chess, after just 19 moves. It was the first time in history a computer had ever beaten a world champion. IBM’s Watson, named after IBM’s first CEO, was developed in 2010, and defeated opponents on Jeopardy in 2011, 100 years after IBM opened its doors. Today, IBM remains an industry leader in technology research and innovation. They have created software for cybersecurity, cloud solutions, and continue to develop Watson’s capabilities to help solve real-world problems (IBM). Mr. Watson Sr. wanted his company to “Think,” and this thought has proven true throughout IBM’s history and continues to flourish today.

**History of Previous IBM Processors**

[2] IBM’s first attempt at building a microprocessor began in 1975, when a team led by John Cocke developed the IBM 801. The project was originally designed to create a telephone switch that would be able to handle one million calls per hour. This idea was ultimately abandoned, but the research on the processor continued. This research would lay the groundwork for future IBM hardware, the IBM Power Architecture family. The design for these processors would be based on reduced instruction set computer principles. This principle was invented by Cocke when originally developing the IBM 801 because he found out that there were larger operations that were slower than smaller operations doing the same task. The IBM 801 project would eventually turn into Project America and Project Cheetah. Project Cheetah would turn into a dead-end, however Project America would end up becoming the IBM RS/600 which would later be named POWER1. The name, POWER, stands for “Performance Optimized With Enhanced RISC” and is the first processor in the line of IBM POWER microprocessors. The POWER1 would be the first microprocessor to use register renaming and out-of-order execution. As time went on newer versions of the POWER series microprocessors would be created with the most recent version being the POWER9. The POWER1 processor was the first one created in 1990 and was built using 32 bits, a clock rate of 20 - 30 MHz,  and 1 core. This same design was used for the POWER2 processor in 1993 as well as POWER3 processor 1998. The only difference between these processors is that IBM began to implement 64 bit processors starting with the POWER3 processor. Every processor that IBM has developed since the POWER3 has implemented 64 bit architecture. The number of cores that each processor has begun to increase starting with the POWER 4 (2001) with 2 cores continuing to the POWER6 (2007) processor. The POWER7 (2010) processor is then upgraded to hold up to 8 cores, the POWER8 (2014) processor increased again with 12 cores, and the POWER9 (2017) processor can now hold up to 24 cores. In addition to the number of cores increasing in each processor, the number of transistors as well as the clock rate for each new processor increased. The first POWER processor began with just a few million transistors while the most recent, POWER9, consists of roughly eight billion. The clock rate of the first IBM POWER processor was about 20 - 30 MHz while the latest processor has a clock rate of about 4 GHz. The line of IBM POWER processors has come a long way since it first began its development. As a result of these processors, we have seen tremendous growth in the way that we are able to compute today. (IBM, 2004)

**Why IBM Power 7 was developed**

[3] As any business does it wants to outdo the competition. Having the best products or being the best provider may guarantee the most profit. This was the case for IBM’s POWER7 when trying to outdo other companies such as HP, Oracle, and etc. The reason for the development was to not only remain as one of the top companies to produce processors but to also increase overall efficiency and productivity.  The POWER7 was marketed from IBM as a processor that can “to manage the most demanding workloads and emerging applications, including a high-end system that offers markedly better energy efficiency than competitive systems” (IBM, 2010). The energy efficiency of the POWER7 was highly marketed to ensure that their processor was the best to purchase and it was something they had over the competition. IBM even made precautions as to undermine the competition by running tests and giving statistics as to why their processor was the best. One statistic was “IBM calculations on the IBM configuration show it requires 35 percent less energy per transaction compared to the energy usage data published by Oracle” (IBM, 2010). To show statistics to one-up the competition helps establish dominance and furthers their likelihood of profit to keep creating new products. At the time before the launch of the POWER7 IBM’s revenue was low and they needed something to increase profits. The POWER7 helped give IBM the spark they needed. With the launch of the POWER7, IBM’s gross profit went from 2.16 percent to 14.56 percent in a year following the launch. This percentage was in the negatives the year before (Macrotrends, 2019). The processor wasn’t entirely made for single users or small businesses. The POWER7 was mainly for “industries that require servers with high uptime, such as the financial or electric industries...The chips are designed for the Internet, database or analytical workloads that process a large number of concurrent transactions, the company said” (Shah, 2010). The overall development of the POWER7 was intended to continue one with new technology and give businesses the tools needed for large workloads to achieve their goals.

**Instruction Set Architecture Overview**

**General Instruction Overview**

[1] The POWER7 instruction set architecture consists of 4 main categories: Base, Server, Embedded, and Miscellaneous Instructions. The ISA specification is split into 5 sections, each of which is called books. The first book, titled *User Instruction Set Architecture,* describes instructions that would be available for application programmers. These instructions primarily fall into the Base category of instruction. Book 1 instructions cover "memory reference flow control, integer, floating-point, numeric, acceleration, application-level programming (Wikipedia). Book 2, or *Virtual Environment Architecture* also covers instructions that are classified as base instructions. These instructions would be used to define the storage model section of application programming and include features such as "timing, synchronization, cache management, storage features, and byte ordering" (Wikipedia).

[1] Book 3 is split into two sections, S, and E. The S section of the book is intended to cover the Server category of instructions. Server instructions are used for a supervisor user for general purpose and server implementations. The second part of Book 3, E, covers supervisor instructions for embedded applications. These types of instructions include "exceptions, interrupts, memory management, debug facilities, and special control functions" (Wikipedia). The last category in the POWER 7 ISA is miscellaneous instructions. These instructions are not partitioned into a specific book, but cover topics like "floating point, vector, signal processing, cache locking decimal floating-point." There is one more book defined in the POWER 7 ISA, which is BOOK VLE (Variable Length Encoded Instruction Architecture). This book is used to define alternates to instructions previously described in books 1 - 3. These instructions all use 16-bit instructions and big-endian byte ordering and are created for low-end applications or for high instruction density. (Wikipedia).

**Overview of Instruction Lengths**

[2][3] The IBM POWER7 processor contains up to eight cores which all have a “64-bit implementation of the IBM Power ISA (Version 2.06 Revision)” (IBM, 2010). The POWER7 has two modes depending on the action being performed. When the POWER7 receives an instruction it goes into the 32-bit mode. This is due to “All instructions are four bytes long and word-aligned” (IBM,  2010). An instruction must be categorized as one following: defined, illegal, or reserved. A defined instruction can either be in the “preferred and/or invalid forms” (IBM, 2010). The defined instructions that use the preferred forms include the Condition Register Logical instructions, Load/Store Multiple instructions, Load/Store String instructions, Or Immediate instruction, and the Move to Condition Register Fields instruction. These instructions in the preferred form will have an efficient instruction execution sequence while other instructions will take much longer to complete execution. An instruction in the invalid form if “one or more fields of the instruction, excluding the opcode field(s), are coded incorrectly in a manner that can be deduced by examining only the instruction encoding” (IBM, 2010). If you attempt to access an instruction in the invalid form the system will then either run its illegal instruction error handler or result in boundedly undefined results. The second class of instructions is the illegal instruction class which includes the set of instructions that have primary opcodes of 1, 5, or 6, primary opcoded that have unused extended opcodes of 4, 19, 30, 31, 56, 5, 58, 59, 60, 62, or 63, and all instructions that are completely made up of binary 0’s. If you try to run an illegal instruction the system will also trigger the illegal instruction error handler and that will be invoked. The final instruction class, reserved instructions, are used to perform different tasks that are outside of the scope of what the Power ISA can process. An instruction is in the reserved instruction class if it has a primary opcode of 0 unless the entire instruction consists of all binary 0’s (this is an illegal instruction), the instruction is not included in the Power ISA (Discontinued Opcodes), the instruction is implementation-specific in order to follow Power ISA specification, or the instruction is implementation-dependent and not described in the Power ISA. Examples of discontinued opcodes: lfq (Load Floating-Point Quad) with primary opcode 56, lfqx (Load Floating-Point Quad Indexed) with primary opcode 31 and extended opcode 791, and stfqx (Store Floating-Point Quad Indexed) with primary opcode 31 and extended opcode 919 (IBM, 2010). If you try to perform a reserved instruction two outcomes will occur: the instruction will be executed if it is implemented in the Power ISA and will trigger the system illegal instruction error handler if the instruction is not implemented in the Power ISA.

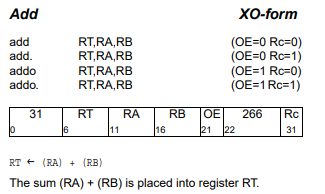


Figure 1: Add Instruction Example (IBM, 2010)

**Instruction Example #1**

[1] One instruction in the POWER7 ISA is an ADD instruction. ADD is an XO-Form Instruction, which means it uses a 9 bit extended opcode. The ADD instruction can be used in four different ways: ADD, ADD., ADDO, and ADDO. The ADDO instructions will require extended arithmetic, and add instructions with the “.” will use the Condition Register and will “reflect the result as a signed quantity” (IBM Knowledge Center). For this example, the ADD instruction is outlined. This 32-bit instruction can be broken down into 7 components. The first 6 bits of the instruction [0:5] are the opcode, which tells the computer that this will be an arithmetic operation. In the POWER7 ISA, if the first 6 bits have a value of 31, (011111), the instruction will be interpreted as an arithmetic XO-Form instruction. Subtraction instructions also have an opcode of 31 in the POWER7 ISA. The next three components in the instruction are the registers. The first register is the RT register, in bits [6:10] of the instruction. The RT register is used as the write-back register, where the result of the addition operation will be stored. In bits [11:15], is the RA register, and in bits [15:20] is register RB. The RA and RB are registers used as the operands, and the values stored in each register will be added together to perform the operation. The next bit in the instruction [21] is the OE bit. The OE bit is the overload exception bit, and is a “field used by XO-form instructions to enable setting OV and SO in the XER” (IBM, 19). This bit will be a 1 if extended arithmetic is needed to complete the operation and a 0 if not. For a regular add instruction, this value will be 0. The next 9 bits of the instruction [22:30] are the extended opcode bits. For all XO-Form ADD operations, these bits will store the value 266 (100001010). The last bit of the instruction [31] is the RC bit. The RC bit is the Record bit, and if it has a value of 0 will “not alter the Condition Register,” (IBM, 20) and if it has a value of 1 it will alter the Condition Register.

**Instruction Example #2**

[2] An example of a store word instruction would contain the following set of bits to determine the instruction. The first 6 bits [0:5] are the opcode and determine the instruction that is to be performed. The RS register is the next 5 bits [6:10] and is where the information will be written to memory. The RA register is bits [11:15] and it holds the current memory address while the last 16 bits [16:31] is the offset. This is an example of what a store word instruction would look in the POWER7 processor. (IBM, 2010)

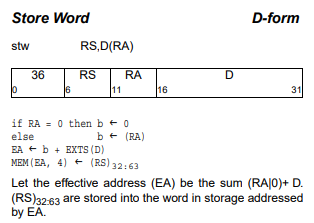


Figure 2: Store Word Instruction Example (IBM, 2010)

**Instruction Example #3**

[3] An example of an addi instruction contains a 32-bit instruction with the opcode being the first six [5:0] bits to tell the processor that it is an addi instruction. The next two sections of the instructions are for registers. The first register in the instruction, when written in assembly language, is the destination RT register or [10:6]. The RA register bits are [15:11] which will contain a value to be added with the immediate. As for the immediate value the POWER7 processor labels this as SI and is in the normal sixteen bits while in assembly language [31:16]. The sum of the RA register value and SI immediate value are placed into the RT register as long as RA does not equal zero. If RA is equal to zero then the immediate value is the only value placed into the RT register (IBM, 2010).

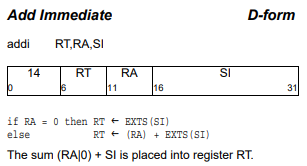


Figure 3: Add Immediate Instruction Example (IBM, 2010)

**Functional Unit Overview**

**Functional Unit**

[1][2][3] In the POWER7 processor, there are 12 execution units. First, there are 2 fixed-point units. These units have a fixed number of digits after the decimal point, and are used for arithmetic calculations. Then there are 2 load-store units, which can also double for simple fixed-point calculations. The load-store units are used for memory access instructions. Next, there are 4 double-precision floating-point units. These 4 units execute double-precision multiply-add operations and can perform 8 floating-point operations per second (FLOPS) per cycle (Sinharoy, n.d.). The double-precision floating point units can also be combined as 2 128-bit vector scalar extension units if needed. The last four functional units in the POWER7 processor are 1 vector unit (Vector Multimedia Extension), 1 branch, 1 condition register, and 1 decimal floating-point unit. The vector multimedia extension (VMX) along with the four floating-point units (FPU) were two separate execution units in the POWER6 processor design. In the updated POWER7 processor the VMX and FPU were combined to form one execution unit called the vector and scalar unit (VSU) in order to reduce power as well as area usage. This new unit integrates the newly designed vector and scalar extension architecture (VSX) which enables the ability to execute “two-way single-instruction multiple-data (SMID) FLOPs. The instructions that the vector and scalar unit would be able to support include VMX/VSX, VMX simple integer, VMX complex integer, VMX/VSX four-way vector single precision, VSX two-way SIMD vector DP, and scalar binary floating-point. The branch unit is used to change the sequence of instruction execution and can be either conditional or unconditional. The condition register unit is used to reflect the result of specific arithmetic operations, and is used simultaneously with the branch unit (IBM, 2010). Finally, the decimal floating-point unit contains instructions that perform “arithmetic, compare, test, quantum-adjustment, conversion, and format operations” (IBM, 2010) for floating-point pairs.

**How the processor works**

**Description of All Pipeline Stages**

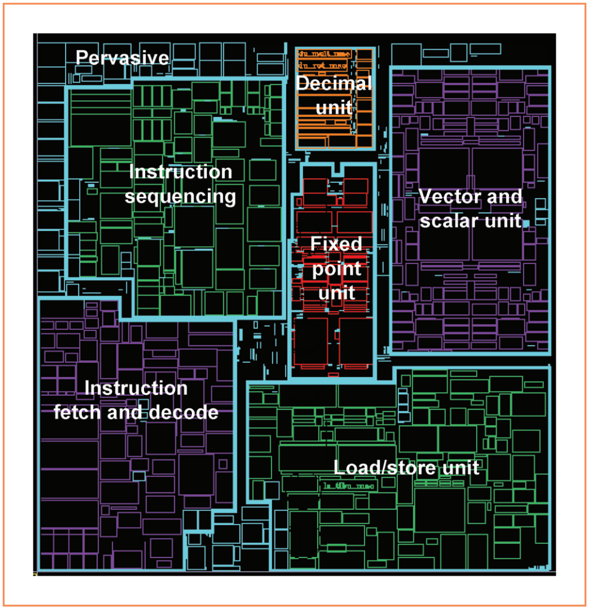
****

Figure 4:  Floorplan (Sinharoy, 2011)

[1][2][3] The POWER7 processor begins its execution of instructions with instruction fetching. The instruction fetching unit will feed instructions into the L1-cache taken from the L2 cache. What is taken from the L2 is an instruction of  “four sectors of 32 bytes each” (Sinharoy, 2011). The instructions fetched can be categorized as the missed L1 I-cache result or instruction prefetches. The requests that the instruction fetches are either prefetch or demand. These requests are independent of each other and are made for all four threads. Two cycles are taken after the instructions are fetched from a subsystem of memory. These cycles help form the pre-decode bits and each of the instruction parities. The pre-decode bits “scan for taken branches, help group formation, and denote several exception cases” (Sinharoy, 2011). If there are branch instructions these are used to develop target addresses when the instruction is scanning for branches. The branch instruction is stored in the L1 cache I-cache with the partial target address. The registers pertaining to instruction fetch address registers keep track of the PC for every thread. To locate the thread being used for the given cycle “thread priority, cache miss pending, IBUF fullness, and thread balancing metrics” are all taken into consideration (Sinharoy, 2011).  For branching instructions, the POWER7 uses a three-cycle branch scan loop. The instruction fetching unit fetches eight instructions, scanning those instructions for predicted taken branches and computes their target address. This scan then identifies the branches as unconditional or taken. If the branch is predicted as taken it is then made available as the next fetch for the thread. After branch prediction, the fetched instructions go to the scan logic for branches and to the instruction buffers. Groups of four instructions are made and are formed from special thread priority. “After group formation, the instructions are either decoded or routed to special microcode hardware that breaks complex instructions into a series of simple internal operations” (Sinharoy, 2011).

[1][2][3] The next path taken for any given instructions is run through the instruction sequencing unit. This unit is used to dispatch instructions, rename registers, issue or complete instructions, and handle exception conditions. The groups previously made from the instruction fetching unit are used for dispatching. Before dispatching the groups, register renaming and queues must be done. “Register renaming is done using the mapper logic before the instructions are placed in the issue queues” (Sinharoy, 2011). After this several registers are renamed by being placed in a  pool of entries. For load and store instructions, tags are assigned to them to help organize the flow. To keep the organization of instructions there are three types of queues in the instruction sequencing unit. These three types of queues are the unified, branch and the condition register queue. An instruction is only ready to be issued when it’s operands are finally available. The issue queues are able to issue eight instructions or less per cycle. “The ISU is responsible to track and complete instructions” (Sinharoy, 2011). The tracking and completing of instructions is done as groups that were created in the instruction fetching unit. The completion of a group is only done when all instructions in a group have executed without failure and finally marked as completed. After it’s been completed the instructions are released so the associated resources can be used by new instructions.

[1][2][3] One of the resources used to complete instructions is the load/store unit. This unit “contains two symmetric LS execution pipelines (LS0 and LS1), each capable to execute a load or a store operation in a cycle” (Sinharoy, 2011). Instructions come to the load/store unit from the oldest operations given priority. The data that comes into LSU can come from the L2 cache, FXU, and the VSU. This data can also come out of the LSU to these units. The pipelines for the LSU “can also execute FX add and logical instructions, allowing more FX execution capability for the POWER7 core” (Sinharoy, 2011). The Load/Store unit contains two reorder queues for both reorder queues. “An SRQ entry is allocated at issue time and deallocated after the completion point when the store is written to the L1 D-cache or sent to the L2 cache” (Sinharoy, 2011). As for the LRQ this queue is meant to watch for hazards that out-of-order loads may be happening. The cache that the load store unit is in conjunction with allows for two load instructions and one store instruction. These are the three ports and based on the instruction there will be a different priority for the instruction to be executed. If it so happens that a load instruction misses the cache line due to a bank conflict then we will “ initiate a cache line reload request to the L2 cache, release the issue queue entry, and create an entry in the LMQ… and also to support the forwarding of the load data to the destination register” (Sinharoy, 2011).

[1][2][3] The fixed point unit (FXU) is used for the arithmetic operations ranging from add to bit select to multiplying. These operations can be done in either F0 or F1 pipelines and each operation requires a specific unit to execute. An arithmetic and logic unit is needed to perform compare, add, subtract, and trap instructions, a bit-select unit (BSU) to execute permute (PM) instructions, and a miscellaneous execution unit (MXU) to perform population, parity, and binary-coded decimal assist instructions (Sinharoy, 2011). The main part of the FXU is a GPR file that contains “four read ports, two supplying operands for the FX pipeline, and two supplying AGEN operands to the attached LSU pipeline” (Sinharoy, 2011). The contents that are held in each GPR file for each pipeline are controlled by the ISU.

[1][2][3] The newly designed vector and scalar unit in the POWER7 processor is able to perform double the amount of floating-point operations per cycle per processor chip compared to the previous processor, the POWER6. It incorporates “four floating-point pipelines to support the dual issue of VSX two-way SIMD vector DP floating-point instructions” (Sinharoy, 2011). Each one of these four floating-point pipelines are capable of performing either “32-bit slice of types 3 and 4 or a 64-bit slice of types 5 and 6” (Sinharoy, 2011). This means that when there is an instruction of type 3 or 4 then the processor will require all four floating-point pipelines in order to process the request on pipe 0. When there are 2 instructions of either type 5 or 6, on the other hand, they can be run at the same time on both pipes 0 and 1 using pairs of floating-point pipelines. The logical execution components that are associated with the vector and scalar unit pipeline include the XS, XC, vector PM, and the floating-point pipelines that perform: 1) scalar binary floating-point, 2) vector single-precision floating-point, and 3) vector DP floating-point. The XS component is the one that handles the vector simple FX calculations. This component has been enhanced in order to bring down the pipeline length from 3 to 2 cycles. The permute (PM) is used to execute permutations in addition to rotating and shifting the bytes/bits given. This is the only component that needs a 128-bit-wide dataflow. The other components either use 4 of the 32-bit dataflows or 2 of the 64-bit dataflows (Sinharoy, 2011).

**Instruction execution example #1 – Add**

[1] One example of a pipelined instruction in the POWER7 architecture is for an R-Type instruction, like add. First, the instruction will be fetched from L2 cache and brought to L1 I- cache by the instruction fetch unit (IFU). The instruction unit will create the group for the instruction to go into for the rest of the instruction pipeline. A group of instructions can have “at most two branch instructions and four nonbranch instructions” (Sinharoy, 2011). The group of instructions will then be decoded and then dispatched. Next, the instruction sequencing unit will dispatch “instructions on a group basis” (Sinharoy, 2011), so the add instruction will be dispatched once all the resources for this group are available for use. If the resources are not available, the instructions in the group will be kept in the dispatch stage. Once the resources are ready, the instruction group will be placed into the queue for instruction execution. Add instructions will be put into the Unified Issue Queue (UQ). The add instruction is then passed to the Fixed-Point Unit (FXU) where it is then put in one of the two pipelines (FX0 or FX1) to execute the instruction. Each pipeline has a GPR file, ALU, CNT, BSU, DIV, MULT, and MXU. The GPR file holds renamed registers and architectured registers for up to two threads. If the add instruction is not dependent on the other instructions around this instruction, then the values will come from the GPR for the calculation. If, however, the instruction needs information from another instruction, the values can be pulled from the load result pipelines (LS1 or LS2) or the FX result bus. The two operands of the add instruction will be put into the two opmux multiplexers. The multiplexors have 5 input options and will choose the correct one for the add instruction and pass them into temporary registers. Next, the values in the temporary registers will be passed to the ALU. The ALU will then perform the add operation and return the result to the final multiplexor. This multiplexor has 3 inputs and will choose the input from the ALU for this add instruction. The value will finally be pushed back to either the GPR or as an input for another arithmetic instruction by the result bus. (Sinharoy, 2011).

**Instruction execution example #2 – Load**

[2] For a load instruction the first thing that is processed is the instruction fetching unit. This unit is responsible for bringing in the best possible set of instructions as well as keeping the execution rates of instructions steady. It also reads and assigns the instructions into groups for the rest of the instruction pipeline. The second step for a load instruction is the instruction sequencing unit which “dispatches instructions, renames registers, issues instructions, completes instructions, and handles exception conditions” (Sinharoy, 2011). For a load instruction the ISU allocates a load tag (LTAG) to it in order to control the load instruction flow. The load tag holds the value for a load-reorder queue when the load instruction was created which will be used later on in the ordering step. The next step for a load instruction has to do with data fetching and is done by the load-store unit. This unit has two identical execution pipelines, LS0 and LS1, which are both able to perform a load instruction in a single cycle. The instructions are all sent to the load store unit randomly but are then categorized based on the age of the operation with the oldest given the highest consideration. The load instruction can be executed in either the LS0 or LS1 pipeline. After the execution step is the ordering step which utilizes the load-reorder-queue (LRQ). This queue keeps an eye on out-of-order loads and keeps an eye out for hazards. A hazard will most likely occur if a “younger load instruction executes out of order before an older load or store instruction to the same address” (Sinharoy, 2011). If a hazard is triggered, the load-reload-queue will perform a flush which essentially eliminates the younger load instructions from the thread while maintaining the rest of the instructions in the thread.

**Instruction execution example #3 – Branch**

[3] If there is a branch instruction that the POWER7 has to execute it begins with the IFU otherwise known as the instruction fetching unit. The instruction is inputted and is taken to the L2 cache and then the L1 cache then finally to the IFU. After the instructions are in the L2 unified cache, and “before the instructions are written into the L1 I-cache. The pre-decode bits are used to scan for taken branches…” (Sinharoy, 2011). The way the IFU scans for branches is with a three-cycle loop. The scan is to find branches and predict their address so the computer may stay ahead. With a three-cycle loop we will have two cycles with no fetch address. If the branch is conditional, that is it is taken. Then the branch will have a certain direction that is obtained using a branch history table. This branch history table is a type of entry array that can be local or global, global having both entry global branch history table and entry global selection. With these tables, the IFU is able to predict all the instructions in a group. Groups consist of eight instructions. There are two ways to predict a branch instruction, indirect and subroutine. These are used to put the branch instructions in a link that allows the computer to seamlessly find the fetch addresses in the ST and SMT2 modes. These modes help predict two branch addresses ahead allowing the computer to “handle a taken branch in every cycle” (Sinharoy, 2011). The instructions will be placed in a link register if it has been predicted to be a conditional branch instruction. This link register is used to “provide the branch target address for the Branch Conditional” (IBM, 2010).  Before the branch instruction goes to the instruction sequencing unit it would have already been classified as conditional and the target address would have been predicted. The instruction sequencing unit or ISU will dispatch the group of instructions and in this case the branch instruction to the branch issue queue. The branch issue queue holds the instruction until the earliest time available and sends the instruction to the branch execution unit. With the branch target address already predicted the instruction can then execute its branch instruction in the instruction fetching unit.

**Implementation Advantages and Disadvantages**

**Advantages of the POWER7 Processor**

[1] The POWER series of processors were designed for the best performance possible, with each model outdoing the last. Each POWER7 processor chip has up to eight processor cores, and each of these cores can run 4 threads simultaneously. This allows POWER7 to run 32 processes at the same time. Compared to the POWER6 chip, the POWER7 has 4 times as many cores, 2 times as many threads per core, and therefore can run 8 times as many tasks at once (Shah, 2010). While each core has an L1 D-Cache and I-Cache and L2 Cache, The POWER7 also has a 32MB eDRAM L3 Cache that is on-chip. Compared to previous models where the L3 cache was off-chip, the on-chip has a 6-time reduction of latency between core and memory (Morgan, 2010). Also, the eDRAM (embedded Dynamic Random Access Memory) technology that IBM used on the POWER7 chip can store “one bit of data using only one transistor and one capacitor instead of the six transistors needed for storing one bit using static RAM.” (Morgan, 2010). Designing the L3 cache using this technology was what allowed the POWER7 chip to have space to house 8 cores. Each of these features on the chip allows for impressive processing speeds that were not available in other processors at the POWER7 peak.

[1] At Rice University, the POWER7 chip was the computer performance solution for cancer research. The POWER7 can handle 18.8 trillion floating-point calculations per second, allowing computationally heavy research to be conducted in fields such as “genomic sequencing, protein folding, drug modeling and simulations of molecular-level interactions in tissues” (Rice University, 2010).

[1] The technology in the POWER7 was ahead of other processors, not only focusing on “raw performance”, but also “intelligent performance” (Shah, 2010).  Not only are POWER7 chips extremely fast, they are also extremely energy efficient. IBM used “Unique Intelligent Energy” technology on the POWER7 chip, which “allows parts of a system to be switched off to reduce power drawn” (Shah, 2010). This feature of the POWER7 chip gave its user a competitive advantage. Organizations that needed to maintain massive data centers could count on the POWER7 chip to handle large inputs of data at quick speeds, ensure high uptime, and save money by saving power on systems that were unnecessarily running.

**Disadvantages of the POWER7 Processor**

[2] One of the biggest disadvantages of the POWER7 processor has to do with the costs of running the system. When compared to servers of competing brands with similar capacity sizes, IBM’s POWER7 cost is consistently higher than its competition. Comparing it to the Intel Xeon Processor E7 v2 the IBM POWER7 processor is far inferior. Intel’s processor has “up to 80% higher performance than IBM POWER7 at up to 80% lower cost” (IT Peer Network). In addition to the higher performance and lower cost of the Xeon E7 v2, it has “69% lower hardware purchase costs, up to 42% lower power consumption under load and 40% under idle, and 16% higher performance at equivalent load with twice the headroom to grow” (IT Peer Network). Along with these technical disadvantages, IBM announced that on September 20, 2019, it plans to suspend all hardware support for the POWER7 series. This means that if you wanted to continue to utilize any of the POWER7 processors then you would have to pay extra if any hardware maintenance were needed. Anybody that is currently operating with A POWER7 processor would have to either make the switch to the newer IBM POWER processor or continue to use the POWER7 and pay a premium if any maintenance is needed. (IBM, 2017) This shows that although the POWER7 processor has many different benefits to choose from there are still drawbacks that could deter potential customers. Still, even with all of these potential deterrents, the IBM POWER series and especially the POWER7 processor are some of the most reliable in the industry.

**Success/Fail Evaluation**

[3] IBM has been one of the leading technology companies for the past few decades. The POWER7 can be considered as a success for multiple reasons. When looking at the advantages versus disadvantages we see the good outweigh the bad. An example of this is the energy efficiency that the IBM POWER7 displays. Although it may have been less cost-efficient than the competition it wasn’t made for the average user and was made for big businesses. To be more expensive for the average user when it was intended for large amounts of computing power would make sense. Whether it is better to buy for the user can be their choice. As for IBM the POWER7 definitely gave the company some money. After the POWER7 was released IBM’s revenue had a steady growth the next year or so averaging around 6 percent with the highest being a 12 percent growth. The actual amount of revenue dollars wise reached a total of 106 billion dollars (Macrotrends, 2019). Taking this into consideration and the fact that the POWER7 beats the competition in a lot of aspects such as energy efficiency and performance  this could conclude that the IBM POWER7 was a success.

**Conclusion**

[1][2][3] With the jump in profit and revenue overall it was in IBM’s best interest to keep improving and expanding on the original POWER7. The POWER7 had many versions made and produced. Some examples of this are the POWER7 8202-E4B, POWER7 8202-E4C and many more. These versions helped increase IBM sales, which led to the introduction of the POWER8. If it weren’t for the POWER7 IBM wouldn’t have been so eager to produce the POWER8 in 2018. The POWER7 processor was the first of its kind. It revolutionized the industry and changed the way companies produced processors. Many processors created after this were designed with ideas from the POWER7 architecture. IBM’s “Think” motto served them well. The POWER7 processor was thinking toward the future, and IBM continues to think of the future with the POWER8 and POWER9 processors.

**References**

Da Cruz, F. (2004, August). *The IBM Automatic Sequence Controlled Calculator.* Retrieved March 17, 2020, from http://www.columbia.edu/cu/computinghistory/mark1.html

Hall, B. (2012, November 23). *Table Of Contents - IBM POWER7 Optimization And Tuning Manual.* Retrieved March 29, 2020, from https://www.manualslib.com/manual/834996/Ibm-POWER7.html?page=5#manual

IBM. (2004, December 22). *Great moments in microprocessor history.* Retrieved March 17, 2020, from<https://www.ibm.com/developerworks/library/pa-microhist/index.html>

IBM. (n.d.). *RISC Architecture*. Retrieved March 17, 2020, from<https://www.ibm.com/ibm/history/ibm100/us/en/icons/risc/>

IBM. (2010, July 23). *Power ISA™ Version 2.06 Revision B.* Retrieved March 29, 2020, from http://kib.kiev.ua/x86docs/POWER/PowerISA\_V2.06B\_V2\_PUBLIC.pdf

IBM. (n.d.). *Thomas Watson*. Retrieved March 17, 2020, from https://www.ibm.com/ibm/history/multimedia/fulldescriptions/think.html

IBM. (2017, September 19). Services withdrawal: Declaration of plan to discontinue lease, rental, and maintenance services for selected machines. Retrieved April 18, 2020, from https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?infotype=an&subtype=ca&appname=gpateam&supplier=877&letternum=ENUSZS17-0034

IBM Knowledge Center. (n.d.). *Instruction Fields*, Retrieved April 9, 2020, from https://www.ibm.com/support/knowledgecenter/ssw\_aix\_71/assembler/idalangref\_inst\_fields.html

Kalla, R., Balaram, S., Starke, W. J., & Floyd, M. (2010, April 12). *Power7: IBM's Next-Generation Server Processor.* Retrieved April 17, 2020, from https://ieeexplore.ieee.org/document/5446247

Macrotrends. (n.d.). *IBM Revenue 2006-2019: IBM*. Retrieved April 17, 2020, from https://www.macrotrends.net/stocks/charts/IBM/ibm/revenue-power7/#gs.48yij7

Madrigal, A. C. (2011, December 29). *IBM's First 100 Years: A Heavily Illustrated Timeline*. Retrieved March 17, 2020, from https://www.theatlantic.com/technology/archive/2011/06/ibms-first-100-years-a-heavily-illustrated-timeline/240502/

Manousopoulos, S., Moreto, M., Gioiosa, R., Koziris, N., & Cazorla, F. J. (2012). *Characterizing Thread Placement in the IBM POWER7 Processor.* Retrieved April 17, 2020, from http://personals.ac.upc.edu/mmoreto/papers/Manousopoulos-IISWC2012.pdf

Morgan, T. (2010, February 8). *POWER7 v POWER6 - it’s all about the cache*. Retrieved April 13, 2020, from https://www.theregister.co.uk/2010/02/08/ibm\_power7\_chip\_launch/

Rice University. (2010, March 2). *IBM donates supercomputer to Rice University for biomedical research collaboration with Texas Medical Center.* Retrieved April 17, 2020, from https://news.rice.edu/2010/03/02/ibm-donates-supercomputer-to-rice-university-for-biomedical-research-collaboration-with-texas-medical-center/

Shah, A., 2010. *IBM Launches Eight-Core POWER7 Processor, Servers*. [online] Network World. Retrieved March 17, 2020, from https://www.networkworld.com/article/2244650/ibm-launches-eight-core-POWER7-processor--servers.html

Shaw, S. (2014, March 12). *Database performance: Oracle on Intel Xeon processor E7 v2 vs IBM POWER7*. Retrieved April 17, 2020, from https://itpeernetwork.intel.com/database-performance-oracle-on-intel-xeon-processor-e7-v2-vs-ibm-power7/#gs.48yij7

Sinharoy, B. (n.d.). *IBM POWER7*. Retrieved April 9, 2020, from https://www.7-cpu.com/cpu/Power7.html

Sinharoy, B., et al. (2011). IBM POWER7 multicore server processor. Retrieved April 17, 2020, from https://pdfs.semanticscholar.org/b284/43d1b145e781edd3a1e332fa6eaa10dade2b.pdf

United States Holocaust Memorial Museum. (n.d.). *The Nuremberg Courtroom*. Retrieved March 17, 2020, from https://encyclopedia.ushmm.org/content/en/article/the-nuremberg-courtroom

Www-03.ibm.com. 2010. *IBM To Deliver New POWER7 Systems To Manage Data-Intensive Applications*. [online] Retrieved March 17, 2020, from https://www-03.ibm.com/press/us/en/pressrelease/32330.wss