One of two major IBM supercomputer efforts in the late 1960s, the Advanced Computer Systems (ACS) project had significantly more ambitious performance goals than the earlier IBM System/360 Model 91 project, and it pioneered many features that became common decades later. Although the project was canceled, it brought many talented engineers to California and contributed to several later developments at IBM and beyond.

Beginning in the 1950s, supercomputers that were acquired by the US national labs were prestige machines for computer manufacturers, or in today’s terms, they were “halo effect” products. Such acquisitions invariably influenced other buyers, so it was beneficial for large manufacturers such as IBM to have a presence.

In the spring of 1962 the Lawrence Radiation Laboratory in California had installed two large Control Data Corporation (CDC) computers, and in July of that year trade publications reported that the lab had additionally contracted with CDC for a new supercomputer. The news of the CDC supercomputer effort came one year after IBM had withdrawn its Stretch supercomputer (IBM 7030) from the market based on disappointing benchmark results at the Los Alamos Scientific Laboratory in New Mexico.

The official announcement of the CDC 6600 supercomputer came in August 1963 and led IBM Chairman and Chief Executive Officer Thomas J. Watson, Jr., to write his famous “including the janitor” memo. In response, a commitment was made at an IBM executive conference in September 1963 to strengthen the high end of IBM’s New Product Line (NPL), then in development.

IBM had already established two post-Stretch high-performance computer projects: Project X and Project Y. Project X was assigned to the IBM Poughkeepsie Laboratory in New York with a design goal of being 10 to 20 times faster than Stretch. The second effort, named Project Y, was assigned to the IBM Thomas J. Watson Research Center, also in New York, with a goal of being 100 times faster than Stretch.

Project X took on an added air of urgency after the September conference. Based on the argument that software costs would be saved if the Project X design was instruction-set compatible with NPL, it became the NPL 604. In 1964, with the announcement of NPL as the IBM System/360 series, the Project X machine was renamed as S/360 Model 92. Approximately three dozen machines in the Model 90 series were eventually built and sold by IBM (using the model numbers 91, 95, and 195).

Project Y was assigned to Jack Bertram’s Experimental Computers and Programming Group, which began to staff up in earnest in late 1963. John Cocke was the first member of the team, and Bertram recruited Fran Allen, Brian Randell, Herb Schorr, and Ed Sussenguth, among others. Schorr developed an initial instruction set and recruited his former student, Lynn Conway, to work on an architectural simulator. Randell has described his Project Y experiences in an earlier issue of the Annals. Conway has made her ACS-related documents available online and has published an account of her ACS experiences.

**Project Relocated to California**

During 1964, sales of the CDC 6600 were strong, and CDC announced a follow-on model 6800 (later to become the 7600).
Watson was not convinced that the S/360 was the best architecture for a supercomputer, and he wanted a group of talented people to design the highest performance machine possible, “unfettered” by the S/360 architecture and the associated S/360 production problems that were then dominating the company.9 In the spring of 1965, he approved a separate IBM supercomputer laboratory for Project Y in California that would be closer to potential customers such as the Livermore and Los Alamos laboratories. There was also hope that a new lab geographically separated from the normal IBM computer development efforts in New York would help mimic the success of the CDC 6600.10

Watson chose Max Paley as the director of the new laboratory, with Bertram and his group forming the core of the architecture team. Paley was a Poughkeepsie veteran who had worked as a product manager for Stretch. Several engineers from Poughkeepsie, including Russ Robelen, were recruited to build the machine. The initial plans called for a 50-member group with a design freeze at the end of 1965, a prototype by mid-1967, and first delivery in 1968. A building on Kifer Road in Sunnyvale, California, was acquired in 1965, and the new laboratory was named IBM Advanced Computing Systems.

Arden House Planning Workshop

A kickoff conference was held in August 1965 at the Arden House in Harriman, New York, featuring talks by Paley, Bertram, and Schorr.11 Other presentations included an analysis of designs from competing manufacturers by Harwood Kolsky and talks by S/360 Model 92 contributors Gene Amdahl and T.C. Chen. Kolsky had worked at Los Alamos for seven years before joining the Stretch effort, and he would later serve as a team member and marketing consultant for ACS. Amdahl was a much admired computer architect who had left IBM when he was not chosen as the leader of the Stretch project but was recruited back to IBM Research in 1960.5

At the conference, the preliminary machine design included an ambitious 10-nanosecond machine cycle time, six to seven circuit levels per machine cycle,12 pipelined arithmetic units, the decoding and issue of multiple instructions per machine cycle, and special treatment of branches with an execute-ahead instruction buffer. The memory hierarchy included

- 64,000-word instruction memory with four-way-interleaving,13
- 125,000-word data memory with 16-way interleaving and 60–80 nanosecond cycle time, and
- 2,000,000-word bulk memory with 64-way interleaving and 250 nanosecond cycle time.

The design was based on 48-bit words with 24-bit half-words and addresses.14 There were 32 48-bit arithmetic registers and 32 24-bit index registers. There was also an emphasis on a “highly compilable” instruction set using 24-bit and 48-bit instructions that we would now recognize as RISC-like.15 The 24-bit instruction format supported register-to-register operations with a 9-bit opcode and three 5-bit register identifiers as well as register-immediate operations with a 9-bit opcode, a 5-bit register identifier, and a 10-bit immediate value. The 48-bit instruction format supported load, store, and branching operations. Floating-point values were represented in one of three formats: 48-bit single precision, 96-bit double precision, or 192-bit extended precision.

Simulations were presented for the performance of a matrix multiply program with various degrees of loop unrolling. Replicating 27 copies of the multiply-accumulate statements within the loop (that is, unrolling by three for each of the three index variables) would give greater than a five times speedup over a standard matrix multiply loop nest as well as an execution rate greater than one instruction per machine cycle.

However, striking a note that would later radically impact the project, Amdahl argued at the kickoff for a S/360-compatible design, which he felt would provide most of the desired performance with only marginally more hardware.11 He further argued that a compatible machine would provide a growth path for Model 92 customers and could leverage the operating systems and compilers developed for the S/360.

The ACS-1

In 1966 the machine design took on the name ACS-1 and kept most of the Project Y attributes, including the 48-bit word size and the large number of registers. The 192-bit extended floating-point format was dropped to reduce the implementation’s complexity.

Bertram had taken over the project’s day-to-day management, and Cocke was placed in charge of the architecture. Schorr managed the architecture team, initially reporting to Cocke. As the project progressed, Schorr took on responsibility for both the architecture.
and the software, reporting directly to Bertram. Robelen headed engineering for the processor, main memory, and cache, and Billy Joe Mooney led the I/O engineering effort. Bob Domenico had initially led the circuits and packaging group, but he soon left and was replaced by Fred Buelow. During 1966, a new building with 38,000 sq. ft. was constructed at 2800 Sand Hill Road in Menlo Park, California, near the Stanford Linear Accelerator, and the project moved there late in the year.

Initial plans had called for the compiler to generate code to manage the transfer of data and programs from a large backing store to a small amount of high-speed silicon memory. Schorr and Dick Arnold reviewed the progress of this approach, and they concluded that the problem was intractable for the compiler and system software. They then adapted a design for single-level store from an experimental machine, and this became a two-way set associative cache. The memory hierarchy was

- a unified cache (called “high speed storage”) of 32,000 to 64,000 words with a cycle time rated at various points in the project between 10 and 50 nanoseconds and
- a main memory of 512,000 words with a cycle time rated at various points between 125 nanoseconds (for thin-film memory) and 375 nanoseconds.

In 1967, the processor design featured an indexing unit, an arithmetic unit, and a sequencing unit. In the same manner as Stretch, the indexing and sequencing units could proceed ahead of the arithmetic unit so that loads and branches could be started early and executed without causing unnecessary delays for the arithmetic unit. The indexing unit also executed the system-related instructions.

The architecture was register-rich, retaining the two large register sets from Project Y and adding 20 control registers varying in size from 11 to 24 bits. Furthermore, the implementation would contain hidden backup registers to allow loads to start early, even while the corresponding architectured registers were still in use. Thus, the indexing unit had an additional set of 31 24-bit backup registers, and the arithmetic unit had an additional set of 31 48-bit registers.

As Randell and Conway have explained, a dynamic instruction scheduling (DIS) scheme was a key part of the processor design. Conway invented a bit-matrix approach to the scheduling control logic as she was developing the simulator. In 1967, the indexing and arithmetic units each had a six-entry instruction buffer from which two ready instructions could be issued out of order per cycle. The sequencing unit could handle one branch per cycle, so the peak instruction issue rate was five per cycle. In 1968, the buffers changed to three and eight entries, respectively, each with the ability to issue three instructions per cycle. Thus, the peak instruction issue rate grew to seven per cycle.

The processor designers estimated that such an aggressive processor could lose half of its performance to the pipeline disruption caused by branching, so they took several steps to reduce the number of branches required at the architectural level and to overlap the computation of branching conditions with other instructions. (Sussenguth gives more details about the ACS-1 design in the “Branching, Prefetching, and Interrupts in the ACS-1” sidebar.)

Using the simulator developed by Conway, performance comparisons were made with the IBM 7090, CDC 6600, and IBM S/360 Model 91 on several benchmarks, including Lagrangian Hydrodynamics Calculation (LHC) and Neutron Diffusion (ND) kernels, which were parts of the workloads typical at the national labs. Simulation results showed that the ACS-1 would be 2,500 times faster than an IBM 7090 on the LHC kernel. ND was a more complicated kernel, with several branch pathways within the main loop. Table 1 shows simulation results for the ND kernel; the values come from a 1967 presentation and a paper by Schorr containing the 1968 results.

In a 1986 interview, George Kennard, the president of IBM’s Systems Development Division during the 1960s, characterized ACS as being established as a single machine effort. However, the ACS group felt from the start that their charter was to develop multiple machines attractive to the national labs. For example, during a 1967 program review of ACS, Phil Dauber reported a sales projection of five ACS-1 machines and that a smaller, slower, but compatible ACS-2 was also being investigated.

**Software**

An aggressive optimizing compiler backend was an integral part of the project, including in-lining, control-flow analysis, basic-block instruction scheduling, global code motion,
and profile-guided optimization. The compiler team included Allen, Cocke, and Jim Beatty, and the compiler was to include frontends for PL/I and an extended dialect of Fortran IV. Allen published several papers on the optimization techniques that the group developed and was instrumental in working with the hardware group in determining the efficacy of proposed processor changes.

The software architecture group was responsible for the initial operating system, which was to be written in an extended version of Fortran IV. The extensions included many of the OS/360 Fortran H extensions as well as memory pointers, bit string and packed word data types, and syntax to directly generate machine operations. Because the machine was being designed in a period of increasing software expectations on the part of the high-performance user community, an extended version of the operating system was contracted to the Computer Sciences Corporation.

In 1967, Schorr estimated over 100,000 lines of Fortran and assembly code would be needed for the operating system and nearly 70,000 lines for the compilers, assembler, and library routines. He estimated the labor costs at $15 million, an amount that was likely much less than the planned investment in the ACS hardware but that was more than 2 percent of IBM’s net earnings for 1967.

Circuits
The processor implementation depended on an aggressive level of integrated circuitry using state-of-the-art ECL circuits. After a request for a quote was sent to the IBM Components Division, Fairchild, Motorola, and Texas Instruments, Motorola was selected as the die fabricator. Because of the time necessary to design, fabricate, and package the circuits, the design effort extended past the end of 1965, and the target introduction date was delayed by two years to the 1970 timeframe. By 1967, the cycle time estimate had been moved from 10 to 12.5 nanoseconds. Buelow, Dan Murphy, and John Zasio describe their experience with the circuits in a companion paper.

Using the circuits available in 1968, Mooney built a sample module called the “precursor” that ran at the originally targeted 10 nanosecond cycle time. The module was a path through a 24-bit adder, designed with five levels of logic per cycle. Each circuit chip held about 22 circuits and could dissipate up to 3 watts. A thermal module was designed to contain hundreds of these chips, with the chips immersed in FC78 liquid fluorocarbon for cooling.

Dissension
Amdahl had relocated from the east coast to California in late 1964 to teach at Stanford, and he was asked by Bob Evans to keep a watch on the ACS project. Amdahl’s opinions about the advantages of compatibility had been disregarded at the 1965 kickoff, but he continued to make the case for S/360 compatibility after the project was established in California in the fall of 1965. Consequently, his relationship with project members soured, especially with Bertram, and by 1967 he was quarantined and shunned because of his continued advocacy. Thus, he had little impact on the design of ACS-1.

Also in 1967, John Earle, a design engineer originally from IBM Poughkeepsie and well-known for designing a high-speed latch for the Model 91, was relieved of his ACS-1 responsibilities by ACS leadership and assigned to Amdahl. Amdahl convinced Earle of the possibility for a S/360-compatible high-end computer, and the two sketched a machine design that became known as the Amdahl-Earle Computer (AEC/360).

In late 1967 and early 1968, Amdahl advanced his arguments for S/360 compatibility and for the AEC/360 design to various levels of IBM management. The arguments he made to Kolsky included the following points:

- Given the data-type and instruction usage in key benchmarks, the machine should be designed around a 32-bit word rather than 48- or 64-bit words.
- The extensive set of registers in ACS-1 accounted for almost half of the circuit count and required one to two extra levels of logic in decoding.
- With a reduction in word size and with fewer registers, each machine cycle could be performed in five levels of logic versus the seven levels required for what he termed a “fat” ACS-1 implementation or six levels for a “lean” implementation. Consequently, the 12.5 nanosecond machine cycle time could be reduced to 8 nanoseconds.
- There was little gain realized by the ACS-1 branching scheme, so the hardware could instead fetch ahead by two
Branching, Prefetching, and Interrupts in the ACS-1

By Edward Sussenguth

The ability to make a decision and alter its behavior gives the computer its power. For example, we ask a computer to sort information or extract data similar to a given pattern, both powerful decision constructs. Without this ability, a computer would simply be a fast calculator or slide-rule.

At the processor level (or assembly-language level) decision-making instructions are variously called branch, transfer, or jump. For simplicity, I use branch here.

Normally, a computer fetches instructions from memory in sequence, one after another. The branch instruction interrupts this flow by switching the sequence to another memory location. We call the “other” location the effective branch address (EBA). In the ACS-1 design, the EBA is either provided in an index register or calculated by adding the contents of an index register to a constant (immediate) field in the instruction.

A branch instruction consists of three parts:

- whether or not the branch is to be taken (the condition),
- the address to which the branch is to be made if successful (the EBA), and
- when the branch is to be taken (the exit point specification).

Most computers combine all these components (test something, calculate the EBA, and branch after this instruction). Thus, the normal idea is that, when a branch occurs in the instruction stream, the result of that branch (that is, to branch or not to branch) occurs immediately following it. However, this is not absolutely necessary, and the three parts may be separated to allow each of the first two parts to be performed earlier.

To explicitly designate where in the instruction stream a branch is to be executed, John Cocke, Brian Randell, Herb Schorr, and I invented a new instruction for ACS-1 called EXIT, and we renamed the branch instruction as BRANCH-AT-EXIT. (This form of branch instruction has also been called “prepare to branch.”)

To implement this instruction, as soon as possible the machine calculates both whether the branch condition is true or not (part 1) and the EBA (part 2), but it will not actually perform the branching until an EXIT instruction is seen in the instruction stream (part 3). Instead, the BRANCH-AT-EXIT instruction causes the processor to record the condition outcome and the EBA in a new entry in an exit table.

For example, for a goto, the EBA can be computed early in the instruction stream using the BRANCH-AT-EXIT. However, when the EBA must be calculated late (as would be done in a multiway case statement), the calculation may be close to the EXIT instruction, thereby losing this advantage.

In the ACS-1 design, instructions between the BRANCH-AT-EXIT and EXIT instructions are executed normally, independent of whether the branch condition is successful or not. This allows the instruction sequencing unit additional time to determine the validity of the branch, calculate its EBA, and prefetch the proper instruction. Then, when the EXIT instruction is encountered, less time is lost in following the desired subsequent instruction path.

The ACS-1 design has 24 condition bits in a condition-code register. These bits represent various conditions such as a result being zero or negative. The BRANCH-AT-EXIT instruction combines any two of these via any of eight Boolean functions to determine the success or failure of the branch. This permits less discontinuity in the instruction stream.

The EXIT instruction causes a change in instruction sequence to the EBA in the first exit table entry that contains a successful condition. There is no change in instruction sequence if all conditions are unsuccessful. In either instance, the execution of the EXIT instruction deletes all current exit table entries so that the next BRANCH-AT-EXIT instruction can begin with an empty exit table.

A variant of the BRANCH-AT-EXIT and EXIT combination is the SKIP instruction, which takes advantage of the ability to easily invalidate specially marked instructions so that they never contend with other instructions. In the ACS-1 design, each instruction contains a “skip” bit. The SKIP instruction has multiple condition-code tests and sets the skipping state for the processor. Subsequent instructions with the skip bit set execute only when the skipping state is false.

Thus, the advantage of the SKIP is that instructions may be entirely omitted in the instruction sequencing unit when the skip condition is true, and thus the disruption caused by a branch is avoided.

To my knowledge, no commercially viable computer has ever implemented BRANCH-AT-EXIT or SKIP. However, several designs have included similar features.
Almost all of today’s computers prefetch instructions. That is, after fetching an instruction from location \( n \), the instruction sequencing unit prefetches the instruction from location \( n + 1 \). More sophisticated sequencing units may use branching history to control the prefetching.

The ASC-1 design includes a hardware facility called prefetch control registers to contain pairs of addresses: the first is the address of the last branch instruction encountered, and the second is the address that was taken as result of that branch, either \( n + 1 \) or EBA. Using this facility, the instruction sequencing unit can place the address of a branch instruction (address \( n \)) in the first register of the pair and either \( n + 1 \) or the EBA in the second register. When the sequencing unit subsequently encounters address \( n \), it will fetch according to the address in the second register. Thus, for a loop, the sequencing unit fetches the wrong instruction only twice, once the first time through the loop and once at the last time, but it fetches correctly on the majority of iterations of the loop.

To extend this prefetching idea, we defined a set of such registers (say four). When a branch is encountered, its address is tested against the four first addresses; if there is a match, its EBA replaces the corresponding second address and the pair is moved to the bottom of the set. If there is no match, its address replaces the first address in the top register pair, and its EBA replaces the corresponding second address; the pair is moved to the bottom of the set. The movement to the bottom ensures that the most recently encountered branches are remembered. The use of a set of prefetch control registers was first introduced by ACS-1, but the concept has been used in other computers as well (for example, the jump trace buffer in the Manchester MU-5).

The ACS-1 design has two ways to handle interrupts, called hard and soft interrupts. An interrupt causes the processor to transfer control to the relevant interrupt handler routine to respond to an event within the system. Hard interrupts were introduced to handle virtual memory page faults, whereas soft interrupts could be used for responding to I/O events.

To process soft interrupts, a specialized, nonprogrammable, branch instruction (an interrupt call, or IC) is inserted into the instruction stream with its EBA set to a predetermined exception-handling location. IC is fully interlocked, and it allows all instructions prior to it to complete but none of the interrupt handler instructions to start early. A companion instruction (an interrupt return, or IR) is provided to resume the interrupted program.

To handle hard interrupts, program execution is immediately stopped and all important internal processor state (such as all registers, instruction stacks, and scheduling matrices) are saved to memory. A special privileged instruction, SCAN, is provided for the operating system to reload the internal data and restart the processor from where it left off.

Our estimates were that soft interrupts would have a variable response latency, from 0.1 to 15 microseconds depending on the number and mix of instructions being executed at the time of the IC insertion, whereas hard interrupts would have a fixed response latency of 2 microseconds.

References and Notes
2. Because instruction fetch is decoupled from instruction issue by the use of the instruction buffer in the instruction sequencing unit and because instruction fetch is designed to run ahead of instruction issue, the functional units will receive a steady stream of unskipped instructions.
3. The 32-bit ARM instruction set and the Itanium processor family are recent examples of instruction predication. For more details, see the predication and branching sections of Smotherman’s website on EPIC architectures: M.K. Smotherman, “Historical Background for EPIC Instruction Set Architectures,” http://people.cs.clemson.edu/~mark/epic.html.

Acknowledgments
I am indebted to the many individuals who contributed to the IBM ACS-1 design, in particular to John Cocke who inspired so many innovative ideas. Fran Allen, Dick Arnold, Phil Dauber, Charlie Freiman, Leo Hasbrouck, Merle Homan, Jake Johnson, Bill Madden, Robert Rew, Brian Randell, Russ Robelen, Don Rozenberg, Herb Schorr, and John Wierzbicki, to name but a few, contributed an hour-to-hour, day-to-day basis, making the ACS project an exciting place to work.
branches to achieve nearly the same performance.\textsuperscript{36}

- A small cache (about 64 Kbytes) could be packaged near the CPU, thereby reducing memory latency.
- A S/360 version could be constructed with about 90,000 circuits, significantly fewer than the 250,000 circuits that ACS-1 engineers were planning,\textsuperscript{37} and would have 1.4 to three times the performance of ACS-1 at 35 percent of the cost.
- The investment in software would be better used to improve OS/360 rather than building a new ACS operating system. Furthermore, much of the compiler optimization work done for ACS-1 was machine-independent and thus applicable to AEC/360 and other S/360 machines.
- A S/360 version would give a growth path to S/360 customers and would be profitable in the marketplace, whereas ACS-1 would not.

### Design “Shootout”

In March 1968, a four-member AEC/360 task force was appointed by IBM management, with Carl Conti, a highly respected individual from IBM Poughkeepsie, serving as chair. The task force accepted Amdahl's estimates of the cycle times for ACS-1 and AEC/360 (that is, 12.5 versus 8 nanoseconds) and compared the estimated performance of the two designs on a set of five kernels (some of which we would today call microbenchmarks).

One of the problems the task force faced was that the ACS group used different benchmarks than those common in S/360 design. The S/360 yardsticks may have been more realistic for general-purpose business and scientific workloads; however, the ACS architects considered the ACS-1 a purely scientific computer, and the benchmarks they used were directly aimed at the national labs. Furthermore, because there was no simulator for the AEC/360 at that point, the performance estimates for the AEC/360 had to be done by hand. Thus, only short instruction sequences could be compared. Finally, there were disagreements about how the short traces of S/360 instructions used for some of the comparisons should best be translated into sequences of ACS-1 instructions.

In a memo describing the comparison, Conti recorded several observations\textsuperscript{38} that included the following:

- AEC/360 excelled on two of the three integer-based kernels with up to a five times advantage.\textsuperscript{39}
- AEC/360 ranged from 2.5 times slower than ACS-1 to slightly faster on the two floating-point kernels, depending upon assumptions made about the code sequences used and whether higher speed floating-point functional units were added to AEC/360.\textsuperscript{40}
- The ACS-1 branching architecture was among three features felt by the task force to provide a 10 to 20 percent performance advantage. However, the task force felt that these three features could either be incorporated into the S/360 architecture as future instruction set extensions or used to motivate implementation schemes that would capture most of the performance advantage.\textsuperscript{41}

<table>
<thead>
<tr>
<th>ND kernel</th>
<th>IBM 7090</th>
<th>CDC 6600</th>
<th>S/360 M91</th>
<th>1967 ACS-1 five-way issue</th>
<th>1968 ACS-1 seven-way issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine cycle time (nanoseconds)</td>
<td>2180</td>
<td>100</td>
<td>60</td>
<td>11</td>
<td>12.5</td>
</tr>
<tr>
<td>Effective cache hit time (machine cycles)</td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Memory access time (machine cycles)</td>
<td>1</td>
<td>10\textsuperscript{23}</td>
<td>13</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>Total instructions for loop</td>
<td>78</td>
<td>81</td>
<td>63</td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>Memory references</td>
<td>139</td>
<td>59</td>
<td>57</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>Total cycles</td>
<td>295</td>
<td>296</td>
<td>148</td>
<td>45</td>
<td>33</td>
</tr>
<tr>
<td>Instructions per cycle</td>
<td>0.26</td>
<td>0.27</td>
<td>0.4</td>
<td>1.3</td>
<td>1.8</td>
</tr>
<tr>
<td>Time (microseconds)</td>
<td>643</td>
<td>31</td>
<td>9</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>Relative performance</td>
<td>1</td>
<td>21</td>
<td>72</td>
<td>1,286</td>
<td>1,608</td>
</tr>
<tr>
<td>Architectural advantage\textsuperscript{24}</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>6.5</td>
<td>9</td>
</tr>
<tr>
<td>Performance limiter</td>
<td>Sequential machine</td>
<td>Instruction fetch</td>
<td>Branches</td>
<td>Arithmetic Circuit speed</td>
<td></td>
</tr>
</tbody>
</table>

### Table 1. Performance estimates for the Neutron Diffusion (ND) kernel.
The longer word length of the ACS-1 was among several features that “either do not affect the CPU performance as measured in kernels or have a negative effect on CPU performance.”

The need for an optimizing compiler was judged as a disadvantage: “Further, it would seem that the importance of the architecture is more than nullified when one considers how dependent performance is upon the algorithm chosen to solve a problem and the quality of the code produced toward that end. It is apparent that the ACS machine organization is more sensitive than the AEC/360 machine organization to the question of code optimization.”

Overall, the task force concluded in favor of AEC/360. The task force noted that ACS-1 would require its own operating system (an arduous task) but praised the ACS-1 for its system organization and I/O handling and recommended that S/360 efforts adopt those advances.

The key issue for the task force appears to be the 8 versus 12.5 nanosecond machine cycle time. A change to 8 nanoseconds would yield, in itself, about a 35 percent performance improvement (ignoring any memory latency problems). No one in the engineering group accepted the AEC/360’s planned 8 nanosecond concept, which was based on the use of five logic levels.

The dispute appears straightforward: ACS-1 went to great lengths and often quite a bit of added hardware to make improvements, each of which were perhaps in the 5 to 10 percent range. If an 8-nanosecond cycle time had actually been attainable (which we don’t believe possible), we believe that the ACS architects would have sacrificed many of the ACS-1 features (branch-at-exit, change multiply to four from three cycles, and so on) to achieve a 35 percent performance improvement.

In the broader context in which the shootout occurred, the various S/360 models were selling well, and all planned processors were S/360 compatible. A Model 95 had just been accepted by NASA at the Goddard Space Flight Center three months earlier (in January 1968), and IBM had described it as “the fastest, most powerful computer now in user operation.”

Thus, Watson’s concern about the S/360 being accepted at the high end had been allayed at least in part. In the more immediate context, Paley’s relationship with Bertram had deteriorated, and Paley had openly sided with Amdahl prior to the shootout.

The task force recommendation in favor of Amdahl-Earle design was accepted by IBM management, and Amdahl was placed in charge of the ACS project. In our opinion, the true rationale for the change was S/360 compatibility, rather than the estimated improvement in machine cycle time and favorable performance comparisons for AEC/360.

Aftermath of “Shootout”
The change in direction had an enormous impact on the ACS personnel. All upper management, eventually including Paley, left and were replaced by Amdahl; most in architecture returned to IBM Research in Yorktown; and many engineers either returned to their original laboratories or formed new companies (many of which became successful).

Among those leaving, Cocke took a sabbatical at the Courant Institute of Mathematical Sciences at New York University to work with Jacob Schwartz, Schorr left to head Computer Science Research at the IBM T.J. Watson Research Center and later published an overview of the ACS-1 hardware and software design, and Conway was fired in 1968 in the midst of a gender transition. Sussenguth also intended to resign but was persuaded by Evans to remain with the project and take the lead of a software architecture team.

Fundamentally there was chaos; due to the numerous departures, the previous day-to-day working relationships were gone and often difficult to replace. However, in Sussenguth’s view, two key players remained: Jake Johnson, who had taken charge of Conway’s CPU simulator and converted it to use S/360 instructions and provide performance estimates for the ACS/360, and Robelen, who was the key engineering interface to determine if ideas were viable and not too expensive in circuitry. (Robelen provides more details on this in the “ACS System Engineering” sidebar.)

The machine design, now termed the ACS-360, had to discard the incompatible branching and instruction skipping schemes. Table 2 compares the ACS-1 and ACS-360, and Sussenguth describes the revised design in detail in the “ACS-360” sidebar. The DIS scheme from the ACS-1 was merged with a register renaming scheme, and this approach was patented by Leo Hasbrouck, Bill Madden, Robert Rew, Sussenguth, and John Wierzbicki. (Because they had left the
ACS System Engineering

By Russell Robelen

ACS System Engineering was the group responsible for the logic design and the hardware debugging. There were separate groups for the architecture, circuits and packaging, and software.

Most of the early ACS System Engineering group came from IBM Poughkeepsie, where they had been involved with the System/360 Models 50, 65, and 75 designs. Many had also worked on the IBM Stretch prior to S/360, and the Stretch experience would prove useful in dealing with the complexity presented by the ACS architectural objectives. Although the history books portray Stretch as a failure, the engineering people at the working level in Poughkeepsie, both those that worked on it and those that did not, felt it was a great success. The engineering was first class. The failure to achieve the cost/performance goals was due to decisions at the top of the project, not the bottom.¹

The S/360 architecture group in Poughkeepsie was divorced from the three engineering groups that were implementing the Models 50, 65, and 75. The mandate of upward and downward compatibility in the S/360 family dictated that this be true. The ability to change the architectural definition, after the fact, to match the hardware of a particular model was not possible in S/360, even though before S/360 this often happened in Poughkeepsie.

At ACS the situation was different. There the engineering groups worked closely with the architecture group in defining the machine. The ability to interact with the programming group was also a new experience for many.

The objective to build the fastest machine possible was exhilarating to the group, and the energy level was high. The sheer size of the machine in terms of circuits was daunting, with estimates up to 330 times the number of circuits in the 7090 and 10 times the number of circuits in Stretch.² For example, the multiplier in the ACS floating-point unit exceeded the circuit count of the entire S/360 Model 50.

The notion of building a pipeline structure with less than 10 levels of logic per stage was also new to many. As one squeezes the number of logic levels per cycle down, the circuit count rises rapidly. This by itself does not necessarily add complexity. The complexity came from the desire to start multiple instructions each cycle looking multiple levels deep into the instruction stream. Furthermore, the goal of a three-cycle multiply was a real challenge.

The implementation of the cache was entirely new, as was multiple fetches per cycle from storage.

The aspect of the design that most worried the engineering team was at the chip level. This was the first implementation within IBM of integrated circuits at the chip level. The realization that one could not make engineering changes to a chip containing a large number of logic circuits meant the design had to have fewer logic design errors than historically had been the case. This

company by that point, Conway and Randell were not included among the inventors.) The register renaming scheme was based on a pool of registers rather than using individual backup registers, which allowed tight loops to be unwound by the hardware, with several independent loads started to the same architecture register. This renaming scheme was later a prominent feature of the IBM RS/6000, along with a limited multiple-instruction issue approach.

Termination

To achieve a profit for ACS-360, Amdahl asked IBM management to approve three models: the high-performance version sketched by Amdahl and Earle, a second version with one-third the performance, and a third version with one-ninth the performance. In May 1969, IBM management rejected Amdahl’s plan for three models, and the project was cancelled.⁴⁸–⁵⁰

After the cancellation, a large number of ACS engineers wanted to stay in California. Several chose to work on disk drive systems at the IBM San Jose facility, including Mooney. Robelen and Cesare Galtieri left IBM to form MASCOR (Multi Access Systems Corp.), and Beebe, Buelow, Zasio, and others left IBM to join MASCOR. Amdahl resigned in September 1970 and formed his own company, the Amdahl Corporation, shortly thereafter.⁴⁸–⁵⁰

Many of the former ACS engineers at MASCOR joined Amdahl after MASCOR closed because it had been unable to obtain the additional venture capital necessary to stay afloat.

The S/360 Model 195 was announced in August 1969 after the ACS cancellation, and a vector-processing task force was started in Poughkeepsie that same month.

Several of the other ideas and technologies developed for the ACS bore fruit in later
kept many members of the team awake at night, and we tried to do a more careful design than might have been done on other machines. It was hard to mitigate the risk, but we were helped by the facts that there were a lot of registers in the machine and that registers tend to be regular and less prone to design errors.

Some members of the team who had worked on the Models 50 and 65 had seen the advantages of the control store and microprogramming technology incorporated in those designs and how it had reduced design errors in the control section of the machines by close to an order of magnitude. However, this technology could not be used in the ACS pipelined design.

As a comparison, the IBM 4341, shipped a decade later in 1979, was the first IBM mainframe computer to ship with integrated circuits. The Poughkeepsie design team was so worried about the consequences of design errors within chips that the first design was mandated to be an exact copy of an earlier shipping S/370 machine. No changes/improvements were allowed. Period. Granted the ACS level of integration at 10 to 30 circuits per chip was considerably less than their 700 circuits per chip of the 4341, but we shared the same kind of worries.

Maintaining acceptable clock skew over 300,000 plus circuits was a challenge. Merle Homan, who worked on high-speed circuit designs for Stretch, designed the clocking system for ACS.

It was easy to get caught up in the euphoria of new concepts and lose sight of the big picture. To many, the lessons of Stretch may have been too easily forgotten.

The later decision to switch the design to S/360 compatibility was a big letdown even though it places many in familiar territory. It was also hard to imagine that IBM would allow a high-end S/360 machine to be built anywhere but Poughkeepsie. The original desire to put ACS far from Poughkeepsie was now gone.

When the project was terminated, none of those from Poughkeepsie were surprised. To many of them, it was the most interesting project they had ever worked on. To many, this may still be true.

Acknowledgments

I would like to give special recognition to the following members of the System Engineering Group: Bruce Beebe, Gerry Paul, Merle Homan, Dick Holleran, Leo Hasbrouck, Bill Madden, and John Wierzbicki. Working with my equal on the I/O side, Billy Joe Mooney, was especially satisfying.

References


work. This includes integrated hardware and optimizing compiler design for the IBM 801 minicomputer, multiple condition codes and register renaming for the IBM RS/6000, and the appearance of DIS schemes in commodity microprocessors in the 1990s. Conway’s work on logic design standardization efforts and the design of the ACS computer-design process also led in part to her work at Xerox PARC on VLSI design and a highly influential textbook with Carver Mead.8

Conway’s collection of documents;7 and Smotherman’s website.52 A video of a presentation on the ACS project given by Sussenguth in honor of Cocke in 1990 is also available online.53

Among the ACS contributors, John Cocke influenced a number of design decisions, from the circuits to the microarchitecture to the instruction set to the compiler. In his ACM Turing Award lecture in 1988, he wrote,

ACS never made it out of the laboratory; I suppose it was too big and expensive, but for me it was probably the most exciting project I have ever been involved in. In reflecting on this, I believe that what made it particularly exciting was that we were a small team, mostly hand-picked by Jack Bertram, and we pioneered every aspect of the project…. My only regret about ACS is that unlike compiler ideas, we did not take the time to publish our ideas on hardware so others could build on them.54
ACS-360

By Edward Sussenguth

The concept of an ACS-360 was simple: System/360 on the outside, and ACS-1 on the inside. S/360 on the outside was straightforward. All S/360 instructions were to be used. The ACS-1 register-to-register instructions, prepare-to-branch, skip, and other instructions were discarded, with the one holdover being start I/O fast release.

However, internally, many of the ACS-1 concepts could be retained, including the separate arithmetic (floating-point) and index (fixed-point) units called A and X.

The A unit included the following:

- two adders (three cycles per operation),
- one multiplier (three cycles),
- one divider (10 cycles),
- one shifter (one cycle), and
- one logic unit (one cycle).

The X unit, on the other hand, include the following:

- two adders (three cycles),
- one multiplier/divider (four cycles for multiply, eight for divide),
- one shifter (one cycle),
- one test unit (one cycle).

Most of the instruction sequencing unit could be retained, including

- a buffer holding instructions,
- issuing four instructions per cycle (two to A and two to X), and
- the prefetch sequence-control registers.

Because of the smaller operand size, the amount of circuitry in adders, multipliers, and so forth of ACS-360 could be reduced. Counteracting this reduction was the need to add the S/360 storage-to-storage and decimal instructions, but the net effect was a circuitry reduction.

Because the S/360 architecture defines only four floating-point registers, a register renaming scheme was developed in which the destination register for each floating-point load instruction is remapped to one of a pool of 16 internal registers.

Branches were handled by adding two one-bit path flags (C1 and C2) to each decoded instruction so that the machine could fetch past up to two conditional branches. First-level-conditional instructions were renamed and sent to the buffers in the A and X units, while second level-conditional instructions were fetched but held at the register-renaming stage. The renaming and instruction prefetching schemes are essentially those used in ACS-1.

Looking back, the register-renaming approach was a good idea, and a similar scheme was used by the IBM RS/6000 some 20 years later.

Another idea used in ACS-360 that appears in retrospect to have been a good one is the addition of a second instruction counter. Observations of the ACS-360 single instruction-counter simulations showed severe underutilization of the expensive functional units on certain problems that had limited instruction-level parallelism due to control and/or data dependencies.

The external definition of a two-instruction-counter machine was that the second instruction counter is equivalent to a second CPU, and all software interlocking and cooperation are considered multiprocessing. Internally, the two programs were separated by “coloring” their instructions either “blue” or “gold” (with a bit added by the hardware) and therefore did not interfere with one another. There were no additional functional units (such as adders or multipliers).

How does one decide whether to issue a blue (B) or gold (G) instruction? The simplest solution is to alternate: B, G, B, G, and so on. However, this has a flaw: if either B or G is idle, a full instruction-issuing cycle is wasted. The algorithm we chose attempted to balance the B and G streams, but it did not allow one or the other to dominate the usage of the functional units. In particular, we chose to throttle an instruction stream if it was causing more than 75 percent utilization of any given function unit.

The two instruction counter idea was not patented because the project was terminated. The idea is now known as simultaneous multithreading and is a common feature in current microprocessors.

See Table 2 in the main text for a comparison of ACS-1 and ACS-360 in terms of architectural and instruction sequencing issues.

Acknowledgments

In particular, Russ Robelen and Jake Johnson ascertained the plausibility of two instruction counters.

Reference

## Acknowledgments

Many people have helped over a number of years to document and explain the ACS and IBM mainframe activities in the late 1960s, including John Adler, Fran Allen, Gene Amdahl, Dave Anderson, Robert Blok, Fred Buelow, Peter Capek, John Cocke, Carl Conti, Lynn Conway, Jim Frego, Charles Freiman, Mike Flynn, Cesare Galtieri, Norm Hardy, Bob Litwiller, Billy Joe Mooney, Ralph Pickett, Jack Powers, Brian Randell, Don Rozenberg, Herb Schorr, Mark Schmalz, Don Senzig, Dick Sites, Julian Thomas, Stuart Tucker, and John Zasio. Special acknowledgments go to Lynn Conway for her collection of ACS documents and to Harwood Kolsky for his collection of ACS documents and detailed handwritten notes, without which we would not have access to many of the details presented here. Dag Spicer at the Computer History Museum was very helpful in providing access to Kolsky's collection. We also made extensive use of the book by Emerson Pugh, Lyle Johnson, and John Palmer and the interviews collected for that effort. Thanks also to Peter Capek, Lynn Conway, Brian Randell, Mark Schmalz, Herb Schorr, and the anonymous reviewers, all of whom provided valuable reviews to earlier drafts of this article.

## References and Notes

1. Now called the Lawrence Livermore National Laboratory.
3. Now called the Los Alamos National Laboratory.
5. For comparison, about 5,000 “medium and large” S/360 computers (Models 40, 50, 65, and 75) had been installed in the U.S. by the end of 1970. Pugh, Johnson, and Palmer, *IBM’s 360 and Early 370 Systems*.
10. The CDC 6600 had been designed by Seymour Cray and a small design team located in Chippewa Falls, Wisconsin, away from the main CDC locations near Minneapolis.


12. This is equivalent to the current terminology of gate delays per clock cycle.

13. The cycle time was left unspecified in the presentation diagrams but was likely planned as 60–80 nanoseconds to match the data memory.

14. A 48-bit word differs from the 32-bit-word approach used in the S/360 but was not an unusual word size for the time. For example, Los Alamos had been using the 48-bit word MANIAC II since 1957, and by 1965, Livermore was using three 48-bit word CDC computers as well as the 60-bit CDC 6600. Both labs also had experience using the 64-bit IBM Stretch. Robert Gregory of the Univ. of Texas computer center remembered, “In April, 1961, a week-long international conference on matrix computations was held at Gatlinburg, Tenn., under the sponsorship of the Oak Ridge National Laboratory and the Society for Industrial and Applied Mathematics. There was unanimous agreement among the participants at that conference that computer manufacturers should be urged to increase the word length on all future scientific computers to at least 48 bits.” R.T. Gregory, “On the Design of the Arithmetic Unit of a Fixed-Word-Length Computer from the Standpoint of Computational Accuracy,” IEEE Trans. Electronic Computers, vol. EC-15, no. 2, 1966, pp. 255–257.


16. Now called the SLAC National Accelerator Laboratory.

17. This is known as a “decoupled access/execute” design.

18. The backup register scheme used a fixed, one-to-one mapping between backup registers and architected registers (except X0 and A0).


20. Although ACS performance goals were originally set with respect to Stretch, Stretch had acquired a reputation for poor performance and commercial failure. Comparisons were instead made to the highly successful 7090.


23. The table entry used in the original comparison represents the CDC 6600 memory cycle time. Timing examples in Thornton’s later book on the CDC 6600 show that memory access time was actually eight cycles. J.E. Thornton, Design of a Computer: The Control Data 6600, Scott, Foresman and Co., 1970.

24. It was defined as relative performance with the difference in machine cycle time factored out.


31. Even though transistor count is the preferred metric of chip complexity today, circuit count was the common metric in the 1960s. A circuit was considered to be a logic gate or flip-flop. Multiply the circuit count by a factor between five and 10 to obtain an approximation to the transistor count.


35. The estimated circuit distribution for the main processor in January 1967 was 42 percent registers, 26 percent function units, and 32 percent control: Anonymous, presentation to IBM Science Advisory Board, 12–13 Jan. 1967.

36. Amdahl estimated that ACS-1 branch preparation required at least seven cycles and argued that most codes would not have enough control-independent instructions to fill the seven cycles.

37. The ACS-1 circuit count estimates varied during the project from 240,000 to 320,000. For comparison, the circuit count for the S/360 Model 91 was approximately 150,000. Schorr, “Design Principles for a High-Performance System,” pp. 165–192.


39. AEC/360 was judged up to five times faster than ACS-1 when considering 20 samples of 30 instructions each from instruction traces for a S/360 assembler. AEC/360 was twice as fast as ACS-1 on list insertion, not only due to the cycle time difference but also because it needed just six instructions versus 24 instructions for ACS-1 for the insertion step (and 127 versus 304 instructions for the complete disk request list insertion kernel). For a short instruction sequence from a radix sort, the times were approximately equal.

40. For a floating-point kernel derived from Livermore’s Coronet II benchmark, AEC/360 needed 66 instructions versus 83 instructions for ACS-1. AEC/360 ran this loop body faster only for the case proposed by Amdahl in which part of the floating-point operations were done in S/360 32-bit single-precision rather than ACS-1 48-bit single precision or S/360 64-bit double precision and in which the AEC/360 was assumed to have higher-speed floating-point functional units. Without allowing the shorter precision or the high-speed units, ACS-1 was estimated to be two and a half times faster than AEC/360. A similar situation occurred with a 15-instruction sequence from an eigenvalue problem.

41. For example, based on efforts at Poughkeepsie, the task force thought that short forward branches, called pseudo-skips in the memo but today called branch hammocks, could be handled within the hardware without instruction set extensions.

42. As a contrast, the CDC 6800/7600 had a projected 25 nanosecond cycle time in December 1964 and was delivered in 1969 with a 27.5 nanosecond cycle time, but like the 6600, the 7600 only issued at most one instruction per cycle. It wasn’t until 1976 with the introduction of the Cray 1 that supercomputers would reach a 12.5 nanosecond cycle time. Even then, the Cray 1 issued at most one instruction per cycle, using six logic levels per cycle. Multiple-instruction-issue microprocessors with short cycle times started appearing in 1989, with the introduction of the IBM RS/6000 and the Intel i960CA, and became commodity items in the 1990s.


44. The IBM press release describes it as a Model 91.

45. The decision can be argued either way based on which set of success metrics is chosen. For example, IBM had been struggling with both OS/360 and TSS software support through 1967 (see Pugh, Johnson, and Palmer, IBM’s 360 and Early 370 Systems), and if a reduction in additional ACS-1 software development and support costs was seen as vital, then the decision was appropriate. However, it is clear that the performance comparison was inadequate by modern standards and that the original goal of an unfettered high-performance machine was sacrificed.


47. The execute instruction in S/360 selects a bit pattern in main memory, modifies subfields.
within the pattern, and then treats the result as the next instruction to execute.


Mark K. Smotherman is an associate professor of computer science and the associate director of the School of Computing at Clemson University. Smotherman has a PhD in computer science from the University of North Carolina at Chapel Hill. He is a senior member of IEEE and a member of ACM. Contact him at mark@clemson.edu.

Edward H. Sussenguth worked in research and development of both hardware and software at IBM, most notably IBM’s networking offerings (known as SNA) of the 1970s and 1980s. He retired from IBM in 1990 and passed away in late 2015. Sussenguth earned a PhD in applied mathematics from Harvard University. He was an IBM Fellow, a member of the National Academy of Engineering, and a recipient of IEEE’s Simon Ramo Medal.

Russell J. Robelen joined the IBM Poughkeepsie laboratory in 1956, where he was involved with various mainframe computers including the IBM 7090 and the Model 50 in the 360 Series. At the ACS laboratory, he managed the design of the ACS-1 CPU and storage control unit. He left IBM in 1969 to cofound MASCOR, and he then spent the remainder of his working career in the venture capital field, retiring in 2002. Robelen has a BS in physics from Rensselaer Polytechnic Institute. Contact him at robelen@gmail.com.

Selected CS articles and columns are also available for free at http://ComputingNow. computer.org.