UCRL-TR-226840

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Early Computing and Its Impact on Lawrence Livermore National Laboratory

W. Lokke

March 21, 2007

This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

December 15, 2006 *UCRL-TR-226840*

Early Computing and Its Impact on Lawrence Livermore National Laboratory

Bill Lokke

How certain notable advances in computing hardware and software affected the uses and users of computing in Lawrence Livermore National Laboratory's early days – 1952 into the 1960s.

The Laboratory began as – and remains – a physics laboratory. Funded by Congress, founded and led by physicists, its mission is dedicated to the application of physics and the supporting science and technology to problems of national significance. And from the start its prime focus has been on the development of nuclear weapons for national defense. Before describing the Laboratory's beginnings, we'll take a look at how it was that the field of physics itself, at the beginning of the century a small, academic discipline, known mostly for its pure research, had by 1952 ascended to such a position of authority in the realms of science, technology and national will.

To many physicists at the turn of the twentieth century, the future of physics looked clear: a few annoying anomalies aside, little pioneering work was needed to explain and predict the unfolding of the natural world. "Modern physics" – the esoteric science of relativity, quantum mechanics, nuclear physics, astrophysics – directly challenged that placid view. Its birth can be marked by the publication in 1905 of several papers by the unknown Albert Einstein, papers that convincingly opened a door to the domains of atoms, quanta, and the consequences of a limit in the cosmos. Purely theoretical and deeply intuitive, this and his later work inspired a rapidly growing number of both theorists and experimentalists to join the expedition exploring these new areas. By the early thirties, this part of physics had been turned into a field of almost non-stop breathtaking theories and discoveries, promising ever deepening knowledge of the very small and the very large, and revealing once again that the unexpected is always upon us.

But it's unlikely that even the wisest seer of the time would have guessed that by mid century, modern physics – and its practitioners – would emerge as the principal driver of critical defense and civilian technology in the Western World; the king of applied science. That in the space of a few years, what was of interest only to academic specialists would so quickly yield such decisive practicality.

This was in large part due to the conclusive event of World War II, the development of the atomic bomb. It had been accomplished by the ideas and hard work of modern physicists working with supporting engineers and scientists in the huge Manhattan Project, in laboratories and newly built plants across the country. The scientific direction for the Project came from a secret laboratory on a mesa in New Mexico, the Los Alamos Scientific Laboratory. And LASL was in turn led by perhaps the leading American theoretical physicist of the day, J. Robert Oppenheimer.

A complex man of many parts – physicist, teacher, esthete– Oppenheimer, for all his brilliance in modern physics, particularly astrophysics, or rather precisely because of it, seemed to lack administrative skills and thus was an odd choice to run the Los Alamos Laboratory. But he surprised his critics by diving into the complex issues surrounding starting up a wartime effort charged with a definite deliverable. The fact that he succeeded brilliantly in providing inspirational leadership to hundreds of extremely diverse individuals simply added to his mystique. When the success of Los Alamos was made public at the end of World War II, he became a national hero.

The development of the atomic bomb had broad implications. For example, nuclear reactors, an application of the new physics needed in the atomic bomb project, held promise to have a similar crucial impact on the civilian sector by producing abundant electrical power, "too cheap to meter". Having made its mark in applied defense work, modern physics and its cohorts were ready to move into the civilian sector.

Then in 1950 President Truman called upon the field of modern physics to develop a new weapon, this time to deter the territorially ambitious Soviet Union: "I have directed the Atomic Energy Commission to continue its work on all forms of atomic weapons, including the so-called hydrogen or super-bomb."[1] The weapon to be developed, the hydrogen bomb, posed greater conceptual difficulties than the atomic bomb. Edward Teller, a Hungarian émigré physicist and veteran of the Manhattan Project at Los Alamos, had spent much of his considerable energy then and since pursuing early concepts without success. But in early 1951 he, along with Stanislas Ulam, proposed a new, more promising approach to the hydrogen bomb. Los Alamos set to work preparing for a multi megaton "proof of principle" test of it in the South Pacific, the Mike shot, planned for October of 1952. But this was truly an experiment, not a deliverable weapon. Many agreed with Teller that such a true deterrent lay too far in the uncertain future.

One of those so concerned was Ernest Lawrence, Nobel Prize winning inventor in 1930 of the subatomic particle accelerator called the cyclotron.[2] After founding his Radiation Laboratory at the University of California's Berkeley campus in 1931, Lawrence attracted a brilliant staff to design and build an ever increasingly powerful set of accelerators destined over the years to reveal many secrets of the atom and, in the process, earn many staff members Nobel Prizes of their own. His team, unique in academia, a multi-disciplinary grouping of physicists, chemists, and engineers working collectively, contributed importantly to both radar development and the Manhattan Project during World War II. In the post war years Lawrence continued to assist the Atomic Energy Commission (AEC) in related projects and worked closely with Teller on defense issues.

To accelerate the development of nuclear weapons, Lawrence and Teller proposed a second weapon design effort, parallel to but independent of Los Alamos. Physicists had earned considerable influence in the postwar world and after considerable discussion in the AEC and Department of Defense, their proposal was agreed to. The laboratory was to be a branch of the University of California Radiation Laboratory to be located in Livermore, 40 miles away from its Berkeley home site.

In the summer of 1952 the task of Lawrence and Teller became planning start-up and scoping the basic support requirements for their new laboratory. At the top of their list came computing – automatic, not human computing. Since the first discussions of a hydrogen bomb, it was obvious that calculating the details of any practical design would overwhelm traditional methods. Human calculations, aided by mechanical calculating machines, were barely adequate for early atomic bomb work. Here the complexities required by the hydrogen bomb concept would defeat a similar approach. What was needed, and needed at once, was an electronic computer.

Electronic computing, as pervasive, powerful and essential as it is today, had barely taken its first steps by 1952. Reviewing its history up to 1952 requires backing up to the early nineteenth century. Mechanizing arithmetical computation began when Charles Babbage created the first mechanical calculator in 1830 – the Difference Engine. Working closely with the daughter of Lord Byron, Ada Lovelace, Babbage moved on to propose the prophetic Analytical Engine, a machine that presaged aspects of the modern stored program computer. But the limitations of high tolerance machining at the time frustrated his construction of a working model. Realization of his ideas would have to wait for the age of electricity. Lady Lovelace's contributions to the concepts of programming such a machine were recognized 150 years later when a new software computer compiler language was named "Ada" in her honor.[3]

Despite these early steps, computing as we know it, as with modern physics, had its foundation laid in the early twentieth century. And again in parallel with modern physics, it occurred in a field held to be notoriously impractical – theoretical mathematics.

Practicality or utility has often been cited as irrelevant success measures of scientific progress – scientific understanding is to be the true goal. The English mathematician G. H. Hardy expressed it this way in 1940.[4]

> "…real mathematics is almost wholly 'useless' (and this is as true of 'applied' as of 'pure' mathematics). It is not possible to justify the life of any genuine professional mathematician on the ground of the 'utility' of his work…The great modern achievements of applied mathematics have been in relativity and quantum mechanics, and these subjects are, at present at any rate, almost as 'useless' as the theory of numbers."

But it was "the theory of numbers", "real mathematics" favorite son, which was fated to provide the underpinnings of one of the century's most useful technologies: modern computing.

In 1900, the great German mathematician, David Hilbert, at the urging of his colleague Hermann Minkowski, presented a paper in Paris that was to inspire generations of future mathematicians. In his paper,[5] Hilbert called on mathematicians in the coming century to take up the challenge presented by 23 unsolved mathematical problems he had selected.

Hilbert chose these problems with several criteria in mind. First, that they represent as wide a range of mathematics as known at the time. Secondly, that their solution would yield new developments in the science of mathematics. Next, that they "be difficult in order to entice us, yet not completely inaccessible, lest [they] mock our efforts." Finally, that collectively their solutions were knowable; the result of what he called "the axiom of the solvability of every problem." Inspiring as his list was, his motivation was similar to the commonly held view in physics at the time: develop and apply our skills and all problems can be solved.

Indeed Hilbert's list became the research agenda for many in the coming years, with the solution of several problems meeting with success. Hilbert, moved by his own belief in universal solvability, later challenged himself and others to develop a complete and consistent axiomatic, logical basis for all of mathematics. In addition, Hilbert hoped to solve the *Entscheidungsproblem*, or "decision problem" by developing an algorithmic method to determine the truth of any logical statement. These broader challenges inspired several more bright minds, including the Hungarian John von Neumann, who became a student of Hilbert in the late 1920s.

However, in 1931 the Austrian Kurt Gödel published a result that severely crippled Hilbert's ambitious axiomatic plan. Encoding mathematical proofs as arithmetic integers, Gödel showed that any mathematical system powerful enough to describe arithmetic was in fact by nature incomplete or inconsistent. The final, crucial assumption of decidability was tackled in a paper[6] read at a 1936 conference by a young English graduate student, the unknown Alan Turing. Stimulated specifically by Hilbert's 10th problem and the *Entscheidungsproblem*, and using formal logic informed by a novel "real world" model of his invention, Turing extended Gödel's results to show that some of these mathematical systems were also undecideable. Ironically, Hilbert, now retired, attended the same conference and on the day before Turing's talk once again declared his faith in the knowability of mathematics, unaware that his dreams were about to be ended.

But it's Turing's theoretical "black box", invented to illuminate the details of his solution, which would have practical impact. His focus on computable numbers in addressing Hilbert's problem required a precise description of the possible processes in computing a number. To that end, he conceived of and developed the required configurations for a virtual, automatic computing machine that would mimic the actions of a human computer following a prescribed series of computational steps, or algorithm, – "without thinking". His theoretical machine is supplied with a paper tape containing "figures", representing both instructions and data. The instructions are coded as individual symbols and the data consists of 0 or 1. The machine can input – "scan" –and output –"print" – one figure at a time. Its memory is enabled by "erasing" and "printing" figures on the paper tape, leaving "rough notes". By defining specific behavior for each symbol the machine can be instructed to calculate a specific sequence. His first reviewer, the American Alonzo Church, who had just published in 1936 a similar result on decidability but using a completely different method, named this the Turing Machine. In his paper, Turing generalized the functionality of these steps to include programmability and showed such a machine capable of calculating any algorithmically computable sequence to any desired accuracy. This construct, which he called the universal machine – the Universal Turing Machine, is at the heart of all current or future computer designs and provides the theoretical foundation of their generality.

Invited by Church, Turing spent the next two years in Princeton (the second year earning his Ph.D.) while several famous mathematicians – Hermann Weyl, Richard Courant, Hardy, and von Neumann – were in residence. Although von Neumann recommended Turing for a fellowship to fund his Ph.D., there is no evidence he had any discussions with Turing about Turing's paper during his time at Princeton. He did learn of Turing's work later and in eight-years time he was to independently write a detailed design description of a buildable Universal Turing Machine – the modern electronic computer.

Turing spent the war years in Bletchley Park, famously providing the ideas behind the machines that cracked the "unbreakable" German Enigma code. He had neither the interest nor the skills in directing projects or applying political influence at any level. As a result, his focus remained on the immediate technical problems. His directness caused some awkward moments in some social situations but he was not a solitary individual. His brilliant contributions to the problems at hand earned the respect of his colleagues. And his success cracking Enigma earned him the direct support of Winston Churchill. Although Bletchley's machines were not Universal Turing Machines, they inspired many to move in that direction, not the least of whom was Turing himself.

During this period the exigencies of wartime and the advances in electronics worked together to advance the state of the art in automatic computing. Scientists and engineers in Germany, England, and the United States, working independently, created a series of machines that were designed to solve specific problems in ballistics, cryptology, and engineering. The one machine arguably taking the longest stride toward the modern computer was the Electronic Numerical Integrator and Computer, the ENIAC.

Commissioned by the United States military for $0.5 million, the ENIAC was built by John Mauchly and J. Presper Eckert at the Moore School of Electrical Engineering in Philadelphia.[7] It was a behemoth. Featuring over 17,000 vacuum tubes, nearly 100,000 other electrical components and weighing 30 tons, it required two and a half years of design and construction. It became operational in late 1945, too late to have direct impact on the war effort. But the ENIAC's performance – 5,000 additions, 357 multiplications or 38 divisions per second – although unimpressive by today's standards, represented for its time a remarkable computing resource. A sequence of calculations requiring a day or two by humans using calculating machines could now be accomplished in less than a minute.

The machine was even more important to the engineers as an existence proof for electronic computing. Common wisdom held that no such large assemblage of perishable vacuum tubes would ever be useful - the mean time to failure for an individual tube would doom such a large assemblage to being inoperative most of the time. In fact, the machine as designed succeeded in far out performing the prediction and put to rest such fears. (However, the basic concern surfaced again later when the technology changed –

the same prediction was made for IBM’s newly transistorized Stretch computer of the early ‘60s. It too lived to refute its gloomy forecast.)

Although designed for the specific task of ballistic calculation for wartime artillery tables, ENIAC included conditional branching and, in principle, could be made to follow any new sequence of computations. This feature was exploited to put the machine to work on other military projects, including some calculations of early hydrogen bomb concepts. Unfortunately, it could only be set up, or “programmed”, by physically changing the wiring pattern in an elaborate wire plug board and by setting by hand hundreds of external switches. The labor involved in setting and resetting these inputs heavily reduced the machine’s effectiveness in moving from one calculational task to another. Well before the ENIAC was finished, Eckert and Mauchly were considering solutions to this problem for a next machine. Then in August, 1944, the peripatetic John von Neumann joined the team.

Throughout the war, von Neumann had employed his considerable skills in a variety of areas: ballistic research for the US Army, the mathematics of explosions and aerodynamics in England, and shock wave calculations for Los Alamos. These experiences made him eager to find ways of improving the necessary calculations. In 1944 he learned of the ENIAC project and was closely associated with the project from then on. In 1945, while the ENIAC was still being constructed, he drafted a detailed plan of a much improved follow-on machine, the EDVAC.[8] In this 35,000 word report, he identified the basic design elements, such as memory and input/output units, needed for a fully automatic computing system. He then developed each in detail, turning frequently to bio-neurological analogs to inform or describe his choices. Never formally published, nor in fact finished – numerous expository footnotes were never written – and referred to as the “First Draft”, it nevertheless was widely circulated and served as the basic blueprint for future computer development.

In a major innovation in computer design, von Neumann’s report suggested replacing the cumbersome external switches and plug board control settings of the ENIAC with digital codes. The codes for a specific set of tasks could be saved on punched cards or tape and loaded and reloaded into memory whenever that calculational sequence – or “program” – was desired. As digits, these instruction codes, along with the data to be computed, would be stored in the same computer memory– or “organ” – as he called it. Also, as digits in memory, they would be as fungible as data and modifiable during the course of the calculation by the program itself. Known today as von Neumann architecture, this graceful solution of a vexing engineering problem thus brought the design back home to Turing’s 1936 paper and the Turing Machine – his elegant synthesis of instructions and data.

The fundamental rhythm of such a computer begins with digits, or bits for a binary machine, being fetched from memory and placed in a portion of a unit called the central processing unit or CPU. The contiguous number of digits fetched is usually set by the designer of the machine, is of fixed length, and is called a word. The word in the CPU is then interpreted as a machine instruction. Instructions, again fixed and selected and implemented by the machine designer can be of many forms. One instruction might fetch another word from a specified memory location and place it in an arithmetic register of the CPU, another might add, multiply or otherwise operate on arithmetic registers of the CPU, still another might logically query one or more such registers.

Except for the last, after completing the action called for by the current instruction, the computer normally repeats the cycle by fetching its next instruction from the next word in memory. Logical queries, such as "Is CPU register A equal to zero?", allow for instruction branching. For example, if a "Yes" result would mean "Select the next instruction from the memory address specified in this instruction," a "No" result would mean "Proceed as normal." Computer designers would have a choice of making many instructions of different types available and building the elaborate hardware circuitry to execute them, or providing a short list, shifting the burden of building the step by step solutions of complex instructions, like taking a square root, onto the programmer.

Since the dawn of the computer age, much has been written of a computer's supposed "intelligence". In fact, as we have seen, the actions of a computer are very simple. Our awe when confronting such a machine stems from its speed in executing its tasks. Whatever "intelligence" that exists in a running computer, and intelligence can be a serious topic of debate, is in the model chosen to be programmed into the machine – playing chess, weather prediction, or sorting through a vast collection of topics. What can be thrilling about running a new program on a fast machine is that results implied by the model but completely unanticipated by the programmer are revealed only through the computer's repeated, simple operations – much like nature herself.

In England at war's end, Turing took up a position at the National Physical Laboratory in late 1945 where he was able to devote full time at last to thinking about building universal machines. By early 1946 he published his "ACE Report",[9] a detailed description of his plans for a stored program computer. His nearly 50 page report fuses his wartime thinking with the latest advances in technology in England. He had seen a copy of von Neumann's "First Draft" the previous summer and refers the reader to it. Turing's and von Neumann's efforts remained separate, following different technological paths. But in 1948 England successfully ran the first program on a stored program computer – the SSEM at the University of Manchester.

Although strongly praised privately by Churchill for his war winning efforts, Turing's place in history as a war hero was still unknown by most. The British government considered the breaking of the Enigma codes "The Ultra Secret" and would wait thirty years before telling its story.

By 1952 both Turing and Oppenheimer, heroes of World War II, were well settled into their respective post war careers. However, both were to suffer devastating setbacks inflicted by the governments they had served. In Oppenheimer's case, his past associations became an increasing security issue with the Atomic Energy Commission, leading some high level officials to express doubts as to his loyalty. Turing, after being victimized by a burglary in 1952, did not hide his homosexuality from the investigating police. Homosexuality being a punishable offence in England at the time, he was arrested, tried and sentenced to a drug treatment "cure" that led to his increasing depression. By early 1954, Oppenheimer's security problems had become front page news. He underwent a government hearing that recommended his clearance be dropped. And in June of that year he was formally stripped of his clearance by the AEC. In the same month, profoundly affected by his enforced drug regimen, Turing ended his life by preparing and eating a cyanide-loaded apple.

Computing in the United States followed closely along the lines outlined in von Neumann's "First Draft". In 1946, Eckert and Mauchly, working with the University of Pennsylvania, began building the EDVAC for the US Army's Aberdeen Proving Ground. However, disagreements with the University over patent rights caused Eckert and Mauchly to leave the project early, taking many senior people with them.[10] The machine finally began running in 1951 but on a limited basis.

In an attempt to bring the EDVAC technology to reality sooner, the National Bureau of Standards created the SEAC (**S**tandards **E**lectronic/**E**astern **A**utomatic **C**omputer). Initially called the National Bureau of Standards Interim Computer, the machine was available in early 1950. Essentially a stripped down EDVAC, it had a simpler instruction set, less memory (mercury delay lines) but the same processing capability: a 0.86 milli-second addition time and 2.9 milli-second multiplication time. The SEAC became the first fully functional stored program computer in the United States.

von Neumann at the Institute for Advanced Study in Princeton worked out further computer ideas with a small team, ideas that lead first, with Nicolas Metropolis, to the Maniac at Los Alamos in early 1952 and then, in 1953, the Johnniac at the Rand Corporation. Additional later versions based on this idea set were built as one-of-a-kind machines elsewhere.

On the commercial front, Mauchly and Eckert created the Eckert-Mauchly Computer Corporation and, in 1946, signed a contract with the Census Bureau for delivery of the Univac 1 – the UNIVersal Automatic Computer. In 1950 Remington Rand bought their financially struggling company, renamed it the Univac Division of Remington Rand and delivered the first Univac to the Census Bureau in March, 1951. Although the Univac represented some performance improvements over the ENIAC, its over-riding advance was the flexibility and adaptability provided by the stored program concept.

So with the spring 1952 AEC announcement of the formation of Project Whitney, the nuclear weapons effort to be located in Livermore, California, Teller, taking von Neumann's advice on the importance of computing for his new laboratory,[11] requested a Univac be ordered even before the official September Livermore Laboratory start date. The production of Univacs was underway and the machines already installed were performing well. The Laboratory's machine, serial #5, would be ready for limited use at the factory in late 1952.

Cecil (Chuck) Leith, a Laboratory physicist/mathematician and a 1943 Berkeley graduate, was to become one of the Univac's first code developers at Livermore. He had first worked with Herb York on uranium isotope separation during World War II at Oak Ridge. While there he made his first computational effort by wiring a plug board on an early IBM accounting machine to perform integration of equations that modeled plant production parameters. Working with York around 1950 he again did some early computational work, this time on an analysis of some early Los Alamos thermonuclear measurements. So when York went to Livermore to head up the Radiation Laboratory's new branch, Leith joined the small group going with him to help things get started in Livermore.

To help publicize the Univac, Remington Rand offered the use of serial #5 to CBS to predict the 1952 presidential election on election eve. The effort required to complete the software program in time was underestimated – as has been the case of most software projects ever since. Completed just in time, the program, using results from preselected precincts, showed by 8:30pm Eastern Time a victory for Eisenhower over Stevenson by a wide margin. Unsure of its accuracy, CBS was reluctant to release the prediction. The network delayed announcing the computer's result for an hour while additional data was entered. The success of the prediction, even with an hour delay, brought the Univac considerable and long lasting favorable publicity.

By September, 1952, Sidney Fernbach[12] was on board to head the computing activities at the Laboratory. Fernbach had graduated in physics from Temple University and entered University of Pennsylvania graduate school before the war. Drafted into the Navy, he was assigned to Japanese language school in Norman, Oklahoma, on his way to serving as a Japanese translator. The war ended before his training was completed and he transferred to the University of California physics graduate school in Berkeley, where he met Edward Teller. He was a first year post doc at Stanford when he received a call from Teller to come to Livermore. Although he, along with almost everyone else, had no experience with computers, with his training in physics he had a sense of what a computer could do and why it was indispensable to the job of nuclear weapon design.

In fact, aside from Teller, almost everyone joined the Laboratory with little prior knowledge of weapons design and computing. From the very beginning, the Laboratory was largely on its own. Although its mission quickly evolved into being a fully competitive second weapons design laboratory, its physics staff – numbering fewer than 20 at its start – was not experienced as weapon designers. No former Los Alamos designers were part of the crew that opened the doors of the Laboratory in September of 1952. The exception, Teller, provided crucial input at the national level and acted as a catalyst in design matters. In fact, hydrogen bomb technology was so new, only some Los Alamos staff members had taken an active role in the design of the thermonuclear portion of the Mike event, scheduled for the fall. Much of that work was done earlier in the year by a team called Project Matterhorn under John Wheeler at Princeton. Some of these people would join the Laboratory in the months to come, notably A. Carl Hausmann, later to lead thermonuclear weapon development at the Laboratory.

What these young physicists did bring to the Laboratory's birth was knowledge of many of the relevant physics equations, knowledge of some of the mathematics to solve them, and an eagerness and confidence to learn, to discover, and apply the missing pieces. To translate this confidence and enthusiasm into action, the Laboratory leaders at its start: Johnny Foster, Harold Brown, and, later, Mark Mills, along with Fernbach, Teller, and Lawrence, joined with its first director, Herb York, to set a strong tone of meritocracy that remained the dominant standard of Laboratory "success" for decades. An employee's value was measured not by external fame or degree status but by what the employee had accomplished at the Laboratory. Initiative was highly regarded and encouraged.

To ensure a good fit at the outset, new employees were encouraged to talk with several groups before picking one to work with. And if later, the employee's interests shifted, the new interest was accommodated. For decades, the prevalent myth throughout Livermore was that no organization chart was drawn for the early days. In fact, a detailed chart drawn in February 1954 has recently surfaced.[13] The fact that the early incumbents denied its existence is some evidence of its irrelevance. People didn't fuss about what box they were or were not in, they focused on the overwhelming job at hand, the job of mastering the art of nuclear weapon design. A climate of urgency and the freedom to create prevailed.

Consider what being a member of this group of new weapon designers would have been like, approaching the task of designing a new atomic or hydrogen bomb, to meet specific criteria. For example, suppose we wish to take advantage of spherical convergence to help create a critical mass of fissionable material. We propose nesting a shell of plutonium inside a shell of high explosive. The proper detonation of the high explosive would send a spherically converging shock inward, collapsing the plutonium into a critical mass.

The physics underlying nuclear weapons technology can be expressed in a series of equations describing the evolution in time of the processes at work. In principle, we can use these equations to answer important questions about our design. Questions such as: How much high explosive is needed to compress the spherical shell of the plutonium to criticality? How much plutonium is needed for a specific yield? Would additional shells of other materials improve the performance? What is the margin of error for a given set of descriptive parameters?

During the Manhattan Project, relatively crude approximations of these fundamental equations were found to be sufficient to design the first nuclear weapons. In some cases, these rough expressions yielded numerical results easily found with the aid of a slide rule or a machine calculator. In other cases, however, the calculations required were complex enough to require computers. In those days, computers were not machines, they were human, usually women. The designer of a particular computation would break the problem into interconnecting parts and assign the parts to the "computers". Much like an assembly line, the output of one or more persons would be passed on to others, who would then perform their assigned set of calculations, pass on their results, and repeat. Usually a mechanical calculating device would have to be used to maintain the number of digits needed for sufficient accuracy. A particular calculation might take hours or days before it was complete.

After World War II the emerging military requirements for nuclear weapons – lower weight, higher yield, sometimes severe constraints (as for artillery projectiles) – demanded a more accurate representation of the physics equations. Often this meant a representation that overwhelmed the capacity of the human computer teams. Under the leadership of von Neumann, Los Alamos began exploring the application of electronic computing to take on these more intensive computing tasks. And from the beginning, any design concept of the hydrogen bomb, with its fusion fuel triggered by an atomic device, a source of intense radiant energy, blast and copious neutrons, was considerably complex and clearly demanded these electronic "giant brains".

Specifically what are these equations, their formulations, and what are the impediments to their solution. To a first approximation, after the high explosive has been detonated, the high pressure conditions existing in a nuclear weapon are sufficient to turn the shocked materials into fluids or near fluids. Much work was done in the eighteenth century by mathematicians such as the Swiss, Leonhard Euler, and a generation later, the Frenchman, Joseph-Louis Lagrange, on formulating fluid flow. With the efforts of succeeding generations of scientists this developed into the discipline known as hydrodynamics. Lagrange developed a generalization such that if the sources of energy are specified and the physical relationships of the materials can be expressed – that is, their pressure as a function of volume and temperature: their equations of state – a concise set of equations predicting the behavior of the material can be explicitly written. Using this result, the Lagrangian differential formulation, in principle the basic method for describing the evolution of material conditions throughout the device – their trajectory in space and time – is in hand.

This technique can be applied to predicting many other phenomena of interest as well: the burning of gasoline in an automobile engine, the evolution of stars or the variability of the weather. Predicting the weather means predicting the time variation of the earth's fluid-like atmosphere. Each spot on earth receives energy from the sun at differing rates over time depending upon latitude, time of day, and season of the year. The atmosphere directly absorbs some of the incident solar energy depending upon the amount of water vapor in the air. The oceans, land, deserts, and vegetation absorb, reflect, and re-radiate differing amounts of sunlight as well as provide quite differing sources for water vapor. As the air temperature rises or falls, its capacity to carry water vapor will rise or fall. The temperature will rise through the release of latent heat when moisture condenses into clouds or rain, fall when the droplets evaporate. Cloud formation alters the energy flow and temperature change by blocking and by reflecting much of the solar radiation, and absorbing and reradiating some of the radiation from the ground back to earth. All these temperature variations result in pressure changes. And these pressure changes, along with the Coriolis and centrifugal forces arising from the rotation of the earth, drive the winds.

However, writing out the basic physics elements of the weather or an exploding nuclear weapon and casting them in Lagrangian form is only the first step. Lagrange's equations may be a good representation of the system being modeled – indeed, these differential equations apply to every mass point in the system. But in the general form, energy sources and equations of state in a complex geometry render a detailed time dependent computation of them impossible. A common solution is to replace the fluid model's smoothly varying differential equations with approximations called finite difference equations. These equations have the effect of reducing space and time to a finite grid of particular values. Space is carved into rectangles in two dimensions or rectangular boxes in three. Time moves in discrete steps, like the ticking of a digital clock. All properties within a subsection or "zone" of space so defined are the same until they change abruptly after a specified time, or "delta T", has passed.

A property, or "variable", such as velocity or temperature, at a particular point in space, or zone, can be "differenced" in differing ways to create a finite difference equation to be solved. For some geometries and physics formulations, the original differential form can be replaced simply by the change or "difference" to be found, divided by the delta T to be advanced, then algebraically solving for the new value of the variable, based on its current value and the various forces working to change it. For other equations, averaging past values of the variable with the current one or sampling neighboring values is necessary. The more complex methods are often used for higher accuracy. However, sometimes they are forced because the simpler ones turn out to be "unstable", that is, they result in an uncontrolled growth of error, destroying the value of any further calculations.

Assuming one has selected a "stable" method of differencing for all the physical properties, the next step is selecting the size of the space zones and time step – the delta T. This is analogous to today's practice of replacing a normal photograph with a digital rendering. In the case of photos, the question is how many pixels are needed to create a satisfactory copy. So too with finite differences: how many zones in space and how short a time step are needed to satisfactorily replace the "analog" differential equations with "digital" finite differences. If the relevant physics has been adequately captured, if the rectangular boxes are small enough, and if the discrete time step is small enough, then the errors generated will be acceptably small. The "digital movie" that results will be a fair predictive picture of the changing weather, the evolving star, or the exploding nuclear weapon.

The first to systematically assemble all the necessary physical processes of a complex phenomenon – in this case, the weather – into a Lagrangian system of equations and attempt a numerical solution was Lewis Fry Richardson, an English meteorologist working during the first two decades of the twentieth century. He reasoned that if sufficient current weather data could be obtained as input and if the equations could be parceled out to enough human calculators, the equations could be advanced in time quickly enough to predict the weather in advance. He actually performed a sample calculation by himself, using data from the morning of May 20, 1910, to predict the change in pressure on a point in Europe six hours later. After six weeks of arduous hand computations, performed during breaks from Richardson's duties as ambulance driver at the front during the dark days of World War I, "My office a heap of hay in a cold rest billet," his results failed to produce an answer close to the measure values – probably because the formalism he chose required calculating small differences between large values, swamping the hoped for results and because he had chosen to reach his prediction time in but one step and that six hour time step was too large for conditions of his calculation. But Richardson was undaunted. In 1922 he published[14] his equations and his results, along with his dream. He envisioned a future “Forecast Factory” larger than London’s Royal Albert Hall, filled with thousands of industrious calculators, with a master conductor orchestrating the flow of results among them, all at work calculating tomorrow’s weather over England and Europe.

Although research in finite difference methods began to accelerate after World War II with the advent of electronic computers, it was just getting underway in earnest by 1952. Similarly, a complete Lagrangian formulation of the physics relevant to calculating a nuclear weapon was still a work in progress at that time. A catalogue of “necessary to consider” factors, similar to those important in predicting the weather, would have to include a broad variety of phenomena. For example, it is important to account for the precise partitioning of energy into the neutrons and nuclei fragments after the fissioning of plutonium or uranium, or into the neutrons, alpha particles, and other light elements after the fusing of materials such as lithium or isotopes of hydrogen. This is necessary in order to account accurately for the subsequent transport of energy as shocks, or x-ray radiation, or high energy neutron transport. Also, materials properties – equations of state, the material’s response to radiant energy transport, neutronic properties – were still being assembled, in some cases even being discovered, and had to be included. And the development of numerical methods for following the absorption and scattering of neutrons as a function of time, the transport of atomic x-ray energy, and complex geometric hydrodynamic flow were all leading edge research topics at the time.

Nevertheless, enough was known to get started, even in primitive form, to make good use of the emerging computing technology. Kenneth Ford, the retired director of the American Institute of Physics, recently recalled[15] using the National Bureau of Standards SEAC to estimate the yield of the first hydrogen bomb test:

> “[In 1952], Princeton's Project Matterhorn took on the task of analyzing the “burning” – or fusion – of the thermonuclear fuel (and the additional fission that it caused). The team at Los Alamos designed the triggering A bomb and analyzed the ignition process. It was my job, aided by John Wheeler and John Toll, to write computer code for the SEAC computer at the National Bureau of Standards in Washington, DC, and then to debug and run successive versions of the program on the graveyard shift (10:00 pm to 8:00 am, if I remember correctly). This program provided the final prediction of the yield of the first thermonuclear explosion, the so-called Mike shot, at Eniwetok on October 31 or November 1 (depending on which side of the date line you were on). That prediction was 7 megatons. The actual yield was 10 megatons. John Wheeler said later that he thought we may have overlooked an energy-generating effect. My own view is that getting as close as we did was a big success, given the primitive nature of the computer (less powerful than today's hand-helds) and the extreme simplifications in the physics that was necessary in order to shoe-horn the calculation into a few kilobytes of code.”

In November, 1952 some of the first staff members from the newly formed Livermore Laboratory traveled to Philadelphia to get acquainted with their still unshipped 13-ton Univac. What they saw bore scant resemblance to what today's notebook computer buyer places on his or her desk. Besides the extraordinary reduction in size, today's computers come with a variety of data entry and output systems: A keyboard that enters information directly into the billion digits of central memory, a screen that can display a full range of colored images, a mouse to alter text or images, ports to a printer or video device, ports to record or load data previously recorded – either on this machine or elsewhere – onto compact disks or memory sticks, and network connections to vast resources of information on the internet.

The Univac input/output systems were a bit cruder. Univac featured a large console with a series of switches that could be set to address each of the machine's 1000 words of memory. Once set, the contents of that memory location would be displayed on the console's oscilloscope. An electric typewriter could be used to direct the machine and was useful for debugging. The largest data system was a set of ten tape units, designed to read and write backwards and forwards. These would serve as an expanded main memory, allowing larger data volumes and even larger programs than would fit in the small amount of main memory to be handled by the machine.

For entering a fresh program or new data, the Univac engineers decided to build a separate machine, called the Unityper. The operator would type information onto a modified typewriter console, wired to transfer the output directly onto metal magnetic tape. The tape would then be loaded into one of Univac's tape drives and its contents read into the Univac memory. Unfortunately, the first version of the Unityper could not signal back to the typist what character had been typed. Having no verification feature made using this system especially challenging; it was like typing with your eyes forever closed. Cecilia Larsen, the first person hired by the Laboratory to prepare input for the Univac, remembers well her first lesson on the Unityper 1 at the Univac factory in Philadelphia: "I learned quickly not to make mistakes."[16] The only method of "checking" the typed input was creating two tapes and comparing one with the other. Larsen and the Univac typists she later hired and trained became legendary in their ability to master the critical function of accurately and swiftly preparing tapes for the many impatient users of the machine.

But there was a better way of entering data. By the 1950s, a well developed and ubiquitous system of recording data was available: the punched card. First developed to control wool weaving looms in the eighteenth and early nineteenth centuries, the punched card idea was adopted by Charles Babbage to control his proposed but never built Analytical Engine. The technology was fully developed in the late nineteenth century by Herman Hollerith, a young inventor and Columbia Ph.D. student. His newly invented card punch and card reader system was selected to record and sort the 1890 census data for the US Census Bureau. The Bureau was very pleased with the results: it reported that the job was completed in a fraction of a year, years more quickly than before and for a cost savings of five million dollars. This success earned his newly formed company contracts for census analysis in several countries around the world, a repeat performance in the US 1900 census, and for Hollerith, his Ph.D. In a few years, his company merged with two others to become CTR, the Computing Tabulating Recording Company. In 1914 Thomas J. Watson, Sr. was named president and in 1924 renamed the company IBM, the International Business Machines Corporation. The company prospered over the next three decades and entered the second half of the century as the world's leading data processing company.

Without punched card input but with the expert help of Larsen and her team, the physicists and mathematicians quickly went to work, writing and debugging computer programs, or "codes", for the Univac. Several code projects were started, each one focusing on one or two of the physics regimes – for example, hydrodynamics or neutron transport. The limited memory of the machine severely restricted both the range of physics issues covered as well as the level of approximation. When the code teams had "finished" their particular code or, in fact long before, the design physicists would begin to use the codes. They would "set up" a problem or "run" by specifying, as input, the geometry and other initial conditions pertinent to the design question being considered – either comparison with other codes or with the limited experimental data at hand. Testing a physics code was not something that could be completed with only one calculation; repeated runs with numerous changes of input parameters were needed to establish a feel for the accuracy range of a particular model. And in the midst of this confirmation process, the new designers were using the codes' results to rapidly field a set of new experimental designs, for both atomic and thermonuclear weapons.

With its 5,200 tubes, miles of wire, and the Laboratory's demanding "always on" schedule, the Univac needed constant care. The Laboratory took the unusual step of hiring and training its own engineers to maintain the machine. This practice was not necessary for most future machines because the vendor would provide their own in-house support staff. Schooled in electrical engineering but usually not schooled in the emerging technology of computer design, the new Laboratory engineering Univac support staff had to quickly learn a new technology "on the job".

Ed Lafranchi's experience was typical. Graduating from Santa Clara University in 1950, Lafranchi became hooked on large scale electronics when working on a Berkeley particle accelerator and, as well, after a stint in the Air Force. Upon his arrival in 1953, he was literally walked inside the Univac, soon to be his home away from home. He shortly became a shift supervisor and faced his first computer crash with considerable unease. "It was late at night and I didn't want to call anybody. My partner and I looked at the printouts and sure enough we found a good old 25L6 – a common vacuum tube – that had gone bad. We replaced it and restarted and away it went. That was it, it was a marvelous feeling: we'd actually figured it out."[17] Before he retired in 1991, Lafranchi was responsible for all electronic engineering at the Laboratory. But after mastering the Univac, his experience with large computers had one more chapter soon to unfold.

After the Univac production was underway, IBM began readying its first entry into the commercial computer industry. From the outset, Thomas J. Watson, Sr. was skeptical of the business value of computers – with IBM owning 90 percent of all tabulating machines, he wanted nothing to diminish IBM's standing at the top of the data processing world. It was his son, Thomas J. Watson, Jr., named company president in 1952, who set IBM on the path of building computers in response to the emerging Cold War. Begun in early 1951 as the "Defense Calculator", the 701 was designed and developed in a remarkably short time. Production began in early 1952 and the machine was formally announced in April, 1952. Serial #1 was shipped from IBM's factory in Poughkeepsie, New York and installed in the Manhattan corporate headquarters at the end of 1952. By April, 1953, the same month as the Univac's installation in Livermore (see the Livermore Computer Acquisition Timeline at the end of this paper), serial #2 of IBM's 701 was operational in Los Alamos. Before the end of 1953, Los Alamos would have two 701s.

As ground breaking as the Univac was, it was soon obvious that it had serious limitations. Chuck Leith had worked swiftly on the Univac and from the earliest days performed all three functions of code building on this and the machines to follow: he developed the physics, worked out the mathematical methods of solution, and programmed the resulting code. His great contribution to Laboratory's fledgling designers was a code for the Univac that could calculate the nuclear and thermonuclear yield of the early designs as accurately as possible at the time. "It was very similar, in many ways, to the kinds of calculations that people were starting in about that time for modeling stars, except that this was dynamic, and those tended to be stationary configurations."[18] But the limited size of the machine made the task quite difficult.

As Leith remembers, the Univac's memory size of 1000 words, each word being capable of representing a ten digit number, originated from an early comment by von Neumann that "a thousand words ought to be enough." Leith says, "We decided later that maybe that was enough for him, but most of us would have preferred a much larger memory." As a concession, the machine did allow for fairly easy buffering from memory to tape and back. So, by moving coding and data in and out the machine in overlays while performing a large continuous calculation, in principle, there was "no particular limitation on the size of the problems that we could carry out." This was facilitated by the tape drive's unique ability to write a tape forward and then read it backwards, a feature that "vanished from sight after the end of that machine." But as all this took very careful coding, few people other than Leith worked to make use of it.

Another difficulty was the actual coding process itself. To write a code for the Univac, one had no choice but to write out in direct machine language the details of each instruction to be executed. For example, to fetch a datum from memory, it was first necessary to state precisely which memory location this instruction was to occupy and then to identify the exact memory address of the datum to be fetched, as well as correctly specify the exact numeric digits for the fetch instruction – any error here and the instruction would be executed as whatever the erroneous digits meant as an instruction to the machine. If in the middle of the coding process, one needed to change the memory location of the datum, every reference to it had to be revised by hand. Awkward enough while still on paper, difficult indeed if it had already been typed on the Unityper. The consequences were that relocating code blocks and generally revising the code was quite time consuming.

The awkwardness and error prone nature of this form of direct computer programming spurred others to develop a solution – the symbolic assembler. The assembler was a program that took advantage of the general purpose abilities of the computer as a Universal Turing Machine. Instead of being primarily concerned with arithmetic, it exploited the branching and logical manipulations in a computer's instruction set to manipulate symbols. In this case, the assembler was coded to associate a symbolic name for each machine instruction and would allow the user to supply an arbitrary name, like "Pi", for the datum. The user could write out his program, not in direct machine language, but using these symbolic names for the desired instructions and memory locations. This code listing would be typed on the Unityper and transferred to magnetic tape. The symbolic assembler, reading the tape, would replace the symbolic names with the actual machine instructions and assign memory locations to the user supplied names. Thus "assembled", the machine-ready code would be written to tape for later use. Symbolic assemblers allowed, among other things, for easier memory management, more understandable coding, and coding in separate blocks without concern for memory overlap. In Leith's view, the introduction of symbolic assembly language was as great a value to the art of programming as perhaps any language innovation to follow.

After a little more than a year of playing a major role in the birth of the Laboratory, the Univac's role as the Laboratory's only computer was coming to an end. The designers demanded more physics in the codes, and more zones with shorter time steps in the runs. And the machine was already oversubscribed – by early 1954, over 60 physicists were working either as designers or code builders. Although the Univac was to spend seven productive years at the Laboratory, it was time to look for the next generation. Fernbach and his team selected the 701 as the next machine in Livermore and 30 mathematicians were now on board to assist in making effective use of the two machines.

On April 9, 1954, serial #16 of IBM's 701 was shipped to Livermore; just three more 701s were to be built. The machine was an improvement over the Laboratory's Univac in all aspects. The 701 was faster – seven times faster additions and four times faster multiplications: 15,000 and 2000 per second[19] and had more memory – 4000 words of electrostatic memory using William tubes, the same technology as used in Turing's Manchester machine. It had 8000 words of auxiliary memory on magnetic drums, four tape drives using oxide coated plastic tape, a printer, and a card punch and card reader. Now a code developer or designer needed just a fresh deck of cards and a key punch machine to prepare input for the computer.

The Williams tubes were deployed because they offered a significant advantage over the mercury acoustic delay lines used in the Univac: The William tubes' memory was randomly accessible to the programmer, thus allowing for a quicker machine cycle time. Each Williams Tube was built around a standard five inch cathode ray tube. Images of bits of memory were painted and refreshed on the phosphor screen of the CRT by its electron gun. Each image would create an electrostatic charge that could be sensed by a metal plate placed on the outside of the tube. Since each tube could image 1024 bits and a word consisted of 36 bits, 36 tubes would be needed to store 1024 words. By spreading out each word across the 36 William tubes, one bit for each William tube, and querying all tubes simultaneously, a full word of memory could be read out in only about 12 microseconds. This compared with a typical 400 microseconds for the Univac memory access time.[20]

Despite its advances, the 701 brought with it a new issue: reliability – or lack thereof. The Univac, anticipating the possibility of machine error, had a variety of checks to ensure accuracy – virtually every computation was repeated and checked automatically by the hardware. The 701 did nothing of the sort, and users quickly discovered that running a calculation more than once and comparing the results was the only way to have confidence in the answers. Already accustomed to a design code's deficiency in physics detail, the resourceful users had to add the possibility of machine error in their evaluations of new computer runs. The William tubes were a significant source of these problems: their erratic performance lead to replacing a tube sometimes every shift. One day, however, the regular mysterious surge in afternoon errors was traced to the late afternoon sun shining on a bank of these tubes whenever someone opened the machine room door. At least this source of the reliability problem was solvable.

With the 701 – even with its troubles – expanding the Laboratory's computing capability, the interplay between code builders and code users increased: their separation was administrative, not in mission. The code builders were in Theoretical, or T Division, with the physicists in one section and the mathematicians in another, the latter under Fernbach. Shortly after the founding of the Laboratory, the weapon designers were separated into two divisions: A (large weapon) and B (small weapon) divisions. Once placed in one of these two divisions, a design physicist tended not to cross over to the other. On the other hand, the boundary between the physics design divisions and T Division was quite permeable, meaning both that some physicists would move between them – sometimes without bothering to formally change their affiliation – and that some codes were developed in the design divisions and some weapons were designed in T Division. Correspondingly, many physicists made significant contributions to both weapon design as well as code development, occasionally at the same time.

The mathematicians in their branch of T Division were to be among the pioneers of a scientific discipline that was not yet named nor for which a system of formal training established. They and their like-minded colleagues elsewhere, particularly in the physics side of T Division, participated in the creation of this discipline by their search for and invention of effective algorithms, efficient ways to store information, software embodiment through programming, and program design for the physics design codes being written. These general activities now form the core of computer science,[21] a discipline not formally declared so until the 1960s.

A particularly polymorphous example of boundary crossing and multi-disciplinary work occurred in T Division in 1955. A small theory group, tasked with designing a particularly challenging new weapon, using a new kind of computer language developed at Livermore, created a weapon design code, termed “Alpha” in this paper, which would have far-reaching utility for future thermonuclear weapon designs.

In 1954, Michael May, later to become a B Division leader and then a Laboratory Director, headed a small group in T Division working on various theoretical models of energy transport. In October, 1954, Roland Herbst, destined also for future leadership – A Division leader, then Associate Director for Nuclear Design, followed by several years of service in the Department of Defense – joined the Laboratory and was assigned to May’s group. Herbst[22] had completed his Ph.D. thesis in general relativity the year before at St. Louis University and, like so many of his colleagues, had joined the Laboratory with no experience in the development of physics codes. May first asked Herbst to transfer a small weapon design code from the Univac to the then new 701. In dealing with the design physicists who struggled to use the code for their design work during the transfer process, Herbst learned the realities of both weapon computer code building and weapon design. During this time May and his group were asked to design a major new thermonuclear weapon – a critical test of Livermore’s design competence. Knowing the features of the evolving design would require new physics not yet existing in any weapons code, he asked Herbst to extend the 701 code to include them.

Spurred by the need to calculate the new weapon, Herbst developed a novel difference scheme for the pertinent differential equations that would remain the “best in class” for decades to come. At about this time May took on other commitments and Herbst was appointed group leader in May’s place. Herbst was joined by William Schultz, hired in May, 1955 with a Ph.D. from Columbia and Norman Hardy, fresh from Berkeley in July, 1955 with a degree in mathematics. Hardy, knowledgeable in physics as well as mathematics, became, as he says,[23] “the mathematical interface between the physics and the code.” Although, as mentioned earlier, a symbolic assembler reduced the burden of developing and programming a large code, such an enterprise was still a major, time consuming project. Since the team had a tight weapon design schedule to meet, the group decided to try and save time by using a new stage in computer aided software writing: the compiler.

A compiler takes the next step beyond symbolic assembly language by providing a language that can interpret limited natural language phrases and the symbols of actual mathematical equations rather than the specifics of machine instructions. For example, consider two lists, or arrays, of numbers, each list for example 100 numbers long. Suppose the task requires multiplying each pair of numbers in the list, A_i and B_i, and storing the result in C_i. In assembly language – symbolic or absolute – the coder would write down, instruction by instruction, the process of copying the number in the first address of A from memory into the computer’s arithmetic unit, fetching the number from the first B address, multiplying it by the number in the arithmetic unit, storing the result in the first C address and sweeping over the loop as many times as there were numbers in the list, in this case 100 times. The goal of a compiler language would be to allow the user instead to write the center of the loop as simply:

```
Do the following as i = 1 to 100
```

$$C_i = A_i * B_i ,$$

with * or X or any convenient character defined in the language to mean multiplication. Many other similar improvements in syntax readability, such as branching statements of the form

```
If X is > Y then Do That
Else Do This
```

can be introduced by a compiler.

After writing the program to be executed in this more abstract language, the user would punch it out on cards and feed it into the computer running the compiler. Much thinking would have to be done by the compiler to translate the user's input into machine instructions. But once finished and written to tape, the final program would be ready for use.

Such a language offers the developer more than increased readability: as an abstraction of the task from the specifics of a particular computer, it can allow a quicker conversion of the user's code to new machines. A major branch of Computer Science today is the study and development of compiler design.

The promise of compilers, breaking through the barrier of complexity faced by the assembly language programmer and moving onto the path of "automatic programming" as compiler advocate Grace Murray Hopper called it, was still largely theoretical. (And, some would argue, has remained so.) Fortran, the first widely used compiler, was in its early planning stages at IBM and not yet named, let alone written. Probably the first demonstration of compiled algebraic equations was written by Laning and Zierler at MIT in the spring of 1954.[24] The compiler Herbst's group used had begun development at about the same time but included features that would make it usable as a rudimentary compiler. Called Kompiler,[25] it was designed, developed and written in Livermore by Kent Elsworth and his collaborators in T Division: Robert Kuhn, Leona Schloss, and Kenneth Tiede. Although in its original form Kompiler lacked the robustness of a complete language, it was designed to efficiently compile the types of arrays important to Herbst's project.[26]

Kuhn was assigned to Herbst's group to assist in expediting the programming of the new physics into the Kompiler language. After intensive work, Kompiler lived up to its general promise of more rapid design code development. The team was able to compile the first version of Alpha on the 701 in 1955 and used the resulting weapon design code to finish the weapon design in time for its successful test. Limited as it was, Kompiler accelerated Alpha's development as well as paving the way for this new way of writing code.

After using Alpha on the 701 to successfully answer the key weapon design question, the group decided to write a better version for IBM's proposed next machine, the 704. Bill

Schultz became responsible for the ultimate code, with Chuck Leith and Frank Adelman joining the team as they prepared for the new machine. Much discussion ensued about the virtues of various differencing schemes, but Herbst remained the overall decision maker.

The planning sessions for the project were particularly intensive. Perhaps a typical session might start in the early afternoon with Schultz proposing a mathematical variant, interrupted by Hardy, saying, “That reminds me of a theorem”, followed by Herbst, famously impatient with obviously muddled thinking by others, protesting loudly – “That’s stupid!” being a signature remark. The others, unfazed, would augment this concert with criticisms and ideas of their own, spilling the session into the evening hours. This not quite subdued, disciplined, civil methodology produced some of the landmark codes to come out of Livermore – in particular the improved version of Alpha, ready to transfer to the 704.

With the success of the 701, IBM had begun a major effort in building its successor, the 704. Besides doubling the speed of ordinary arithmetic operations over the 701, the new machine was to have a number of important improvements, such as index registers and built in floating point.[27] Index registers made unnecessary the repeated altering of instruction addresses in looping the same calculation over a large number of different memory locations. Instead, the programmer would simply tag the memory reference with the register address and the hardware would automatically increment the memory address by the register amount. Floating point was even a greater time saver, both in programming effort and in execution speed. With both the Univac and the 701 offering only fixed point arithmetic, the programmer had to be responsible for ensuring the range of values in a computation would not exceed the limits of the word length, usually the equivalent of 10 digits or so. In scientific calculations, where exponent swings of many factors of ten are common, this could usually be accommodated only by employing software that kept track of a number’s most significant digits separate from its exponent, allowing the latter to ‘float’ in response to operands of widely varying size. By incorporating these floating point techniques directly into hardware, the programmer’s task was greatly eased and the resulting code speeded up considerately. In fact, the 704’s 5000 floating point multiplies per second was somewhat faster than its fixed point multiply speed.

Perhaps the greatest improvement incorporated into the 704 was its introduction of core memory. Co-invented in 1949 by An Wang, later to found the computer manufacturer Wang Laboratories, core memory provided a faster, cheaper, and more reliable form of storage than the Williams Tube. Each bit of a memory word was represented by a small doughnut shaped ferrite core with a network of four wires passing through the hole in the doughnut. A ferrite core could be placed in one of two magnetic states by the passage of suitable current through its center. Two wires would be required to uniquely set one such core in a two dimensional array of cores; the other two wires were needed to read – and reset if necessary – a core’s state. The ability to reset the core was required because the act of reading always forced a core into the same magnetic state; reading effectively erased the memory.

Core memory became the enduring memory technology of choice for generations of new machines, from several manufacturers. The fact that memory is retained – the magnetic states do not change – when power is turned off, made core memory an excellent choice in portable systems. For example, NASA's Space Shuttle computers were built with core memory for decades and weren't replaced with silicon until 1990. The core memory from Challenger's computer memory system was recovered from the ocean's floor intact and read out after the 1986 fatal accident.[28]

Even in core memory's first application in the 704, its cost was attractive and would drop when subsequently assembly plants were built in developing countries. Initially costing about a dollar a bit, by the time silicon chips replaced core in most applications, the cost had dropped to a penny a bit. However, even at that price, modern computer memories using core would not be practical: A computer using core equal to the capacity of the silicon memory in a small PC of today would cost an unmarketable 100 million dollars. Core does live on to this day at least in terminology: the term "core dump" is still used to refer to recording the contents of computer memory on an output device. But perhaps that residual usage is not enough to excuse the wizened programmer of yesteryear from asking how much core is in your notebook.

While the 704 was being developed, IBM agreed to support an effort, initiated and led by the IBM innovative software developer John Backus, to provide an efficient compiler with the machine. Backus received the go ahead from IBM in late 1953. He began by hiring two bright IBM staffers: Irving Ziller, a graduate of Brooklyn College and with the company for only a year, and Harlan Herrick, a five year IBM programming veteran, skeptical that the project could produce anything close to the efficiency of coding like his own. In the spring of 1954, Backus visited MIT to discuss the Laning and Zierler compiler recently unveiled. "It was pretty good, very nicely done conceptually," Ziller recalled. "But they took an academic approach. They couldn't care less about efficiency."[29] In fact the coding produced was 10 times less efficient than a programmer would produce. And Backus's primary, single minded focus was efficiency – the speed obtained by the machine code Fortran compiled. He reasoned that only a truly efficient compiler would ever be used.

This three-person team spent the year developing their ideas and in November of 1954 put to paper their project's goals, language, and name: "Preliminary Report: Specifications for the IBM Mathematical FORmula TRANslating System"[30] – Fortran. Armed with their prospectus, they traveled across the country, visiting sites that had already ordered a 704, looking to spark interest in their compiler. Little was generated. Nearly all thought Fortran was, at best, premature, that since other fledgling projects had produced nothing close to "automatic programming", theirs would not either and could be ignored.

This negative reception had the effect of inspiring the team to succeed and in early 1955 Backus began adding to his team. His group grew eventually to 13 and included mostly young IBM recruits with raw talent who, like Backus himself, were still in the early stages of learning and inventing computer science. Two of the team came from outside the company: Roy Nutt, from United Aircraft, and Robert Hughes from Livermore. Backus had approached Fernbach in late 1955 for a Laboratory member and Fernbach, having supported the Kompiler effort and being encouraged by Backus' plan, suggested Hughes. Hughes, with a master's degree in mathematics from the University of Michigan, had been at the Laboratory for more than a year, working with a physics group on programming a statistical method for solving neutron transport. Like so many others, Hughes took on the sudden assignment to a completely new field with equanimity: "I had learned that if you don't know how to do something you say, 'Yes.'"[31]

Hughes joined the group for the summer of 1956. Initially skeptical, after three months of working with Backus' talented team, "I became a compiler advocate." Knowing that he was unfamiliar with their ideas, he learned their schemes by flow charting the group members' assembly language coding, providing them with useful programming aids. He then moved on to helping the group with the overall documentation of Fortran.

Hughes says a particular conversation with Backus stands out in his memory. Hughes had pointed out what he considered a severe flaw in the language: there was no way to provide user defined subprograms. Livermore codes, being usually quite large, were often written in modular form. That allowed for reusable code blocks as well as memory overlays, the latter allowing small computer memories to execute more code than would fit all at once. Backus replied that his focus was solely on proving they could write a complier that could produce coding rivaling the efficiency of hand programming. With that established, they would move on to add other features, like subroutines.

Like so many other pioneering projects, Fortran's release date kept slipping. When asked, Backus would say, "Six months, come back in six months. We honestly always felt that we were going to be done in six months." Fortran I was released in early 1957, along with its programming manual,[32] where on the title page, Backus listed the thirteen members of his working group, including Livermore's Hughes. From its first release Fortran produced coding of high efficiency, impressing both users and the compiler's own developers with the compiler's cleverness. Hardy later observed that it was ten years before other compilers approached the high level of efficiency that Fortran achieved: Backus and his team had succeeded brilliantly.

The Fortran team continued to produce improvements, with subroutines being featured in the release of Fortran II in 1958. In later years, much was made of Fortran's supposed language flaws, such as the GoTo statement, notorious because of its potential to launch a bug filled code out into unknown parts of memory. Dismay over this statement led structured programming advocate, Edsger Dijkstra, to write a now famous letter in 1968 to a leading computer journal that ran under the title, "Go To Statement Considered Harmful."[33]

The real test of a language, as Backus had observed, is whether it is used. Although it took some time, within a few years Fortran had changed the landscape of programming at Livermore and elsewhere. All did not go smoothly. The weapon design code Alpha team decided to switch from Kompiler to Fortran I, wrote out the revised code in the Fortran language, and attempted to compile it on the 704 soon after its arrival. Unfortunately, Fortran's first release was not up to the complexity of Alpha. Fortran would seem to become stuck, furiously writing and reading its working tapes and drum memory, gamely attempting to compile the Alpha program. Eventually a machine failure or tape problem would interrupt it, leaving the group no choice but to try again. (The 704 was more reliable than the 701, but the reliability rating had only risen from the 701's "truly flakey" to the 704's "quite fallible". It would be another generation of computer design before speed and reliability would come together.) The group decided to switch back to assembly language on the 704, the rewrite greatly speeded by working from the Fortran language. Alpha, continuously improved by Bill Schultz, became a flagship code, unique to Livermore. Alpha, the first Livermore production code ever to successfully exploit the development time saving advantage of a compiler, was moved, ironically often in assembly language, to every new machine the Laboratory was to buy over the next four decades.

For most, "Fortraning" became the fashion. In 1958 the first production design code written in Fortran at Livermore was released from T Division, quickly grew in usage, and evolved to become a dominant design code for many years. Fortran meant the coding was easier to produce, debug, and understand, allowing teams of physicists to work directly on the code and produce improvements very quickly. And as new machines were added to the Laboratory's growing stable of computers, moving codes written in Fortran to the new machine could often be accomplished in days. All that was necessary was a Fortran compiler written for the new computer.

The final proof of Fortran's triumph locally came when Sam Mendocino[34] and other members of Livermore's growing computer language group, having moved on from Kompiler, realized that a Livermore version of Fortran, to be called LRLTran, would best be written not in assembly language, particular to one machine, but – in Fortran itself. Backus went on to make seminal contributions to the science of compiler design, and in 1977, received the Association of Computing Machinery's Turing award – computer science's equivalent of the Nobel Prize.

The 704 was to become a commercial success, with 123 sales during its market lifetime of 1955 to 1960.[35] Announced in May, 1955, the first 704 was installed at IBM headquarters in December of that year. Within two months, Livermore received its first 704 in February 1956, two months before Los Alamos took delivery of their first machine. In July, a second 704 was operational at the Laboratory; two more would find their way to Livermore: the next eleven months later, the last arriving in August 1958. Fernbach had hit his stride in buying computers.

As the arc of a person's life is often set by behavior evidenced at a very early age, or in the extreme, like the universe itself, whose subsequent 13.7 billion year trajectory is the outcome of its first 10^{-26} seconds, so to at the Laboratory, by the time the 704s were being used regularly, designers and code developers had established specific patterns of expectations and behaviors that were to define the Livermore approach to nuclear weapon design for the rest of the Cold War. At the heart of the approach was the axiom that designs must be thoroughly computationally modeled before being detonated in the Nevada or South Pacific test sites. Since the arrival of the Univac, and accelerated by the availability of the 701 and the 704, the goal of the theoreticians and code builders was to create a set of physics design codes that, through ever improved approximations and repeated testing and comparison with experiments, would be as complete and accurate a set of tools in the hands of the designers as possible. As the early charter for the Theoretical Division stated, theoreticians were to provide "theoretical verification by computation." And the thoughtful designers learned early in their careers that the path to success was through the careful, thorough use of this tool set, examining each aspect of the design – particularly as a check on the designer's intuition and slide rule.

As computing capacity began to rise, competing codes for similar physical processes were encouraged – or, as often, allowed. If another method of differencing the equations for a process already represented in an existing code seemed more promising, or another level of approximation appeared to be an improvement, it would be developed and added to the mix of codes. The designers in turn would use the new code to compare with the existing codes on test problems, previous experiments, and with caution, on new designs. If the new code was accepted by at least some users, it would take its place in the mosaic of codes covering design space. In this way, "good codes would drive out bad codes" and this Gresham's Law in reverse would continue to raise the quality of the code tool set to ever higher levels.

A striking feature of this practice was the common belief that such a goal was attainable. On its surface, this was a bold claim for several reasons. Even for fairly crude designs, the computer limitations of speed and memory severally circumscribed the extent of detail any code could represent. For decades to come, these same limitations would prevent any one code from incorporating more than a few of the physical processes necessary to completely model a weapon. And finally, it was debatable whether each of the complex physical processes involved could ever be captured sufficiently fully and accurately.

Think of attempting to grasp the complete behavior of a nuclear weapon design as a variant of the classic "elephant in the room" problem. In this case, comprehension and analysis is desired but the problem is too big to be solved in one glance – and the light is dim. The designers must rely on their judgment on how to use the codes to their best advantage. Which codes from the mosaic to use, what features of the design to parameterize, how many individual runs are needed, which codes to compare, and perhaps most importantly, what exactly to look for in all these results – these were the kinds of decisions, enacted repeatedly, often over many months, that would determine the fate of a design. The review process by the designers' peers was frequent and often intense, but came after much work was done. The engineers and chemists were more than important: they created a physical object from drawings and specifications. The test program people brought about and diagnosed an experiment, providing precious data about the design. But it was clear: the fidelity of the codes and the designers' judicious use of them were at the front of the chain and thereby were the keys to success or failure.

This challenging goal was a huge inspiration to the young imaginations of those early days. After a rocky start, designs were created and tested that lifted the sophistication of the designers beyond that gained by experiments alone. Creative computer studies of the systematics of design or careful analysis of unexpected patterns in certain design calculations would lead to fruitful insights in understanding existing weapons or spark new concepts. Designers were finding computing to be more than "theoretical verification by computation"; it was also a source of discovery.

For most designers and code developers, working with computers meant direct interaction with the machine. But getting on a computer was never as easy as today's clicking "Start" on a PC. For starters, a crew working 8-hour shifts, 24 hours a day, 7 days a week was needed to keep the complex hardware of the computer running round the clock. Then, at least for machines after the Univac, a user would carefully prepare for his or her time slot by having a deck of IBM cards punched, often by himself or herself, containing the information to be run. For designers, the deck would contain the information needed to specify the starting conditions for a particular code run. For code developers on machines during much of the 1950s, the deck would have to include all the instructions needed to bring the machine up from an idle condition – a computer "operating system" was a later development.

Usually the deck was given to the machine operator who would hang whatever tapes – half inch wide, 1400 feet long, open reel, a foot in diameter – into six-foot-tall tape units that the "job" would need, load the input deck into the card reader, and hit start on the card reader to begin operation. Typically the first few cards in the deck instructed the machine how to read the rest of the deck or would signal the machine to load a preexisting program from a standard tape unit.

During prime daytime hours, the machine time was usually allocated to the testing and debugging of the physics design codes. The time allocated to a code team, called a "debug shot", was usually used by the code team to either run a brief test of some newly developed section or check only to see if it compiled or assembled properly. The wise developers were on hand during the shot, ready to leap to a nearby key punch to make any corrections if needed. If the cause of any trouble was not obvious, or if the user had accidentally submitted a faulty deck to the machine operator without attending the run, the machine slot was lost, and the next job was loaded. Only the often enigmatic print out of the event was evidence to aid in preparing for the next attempt. Because the next attempt might not be until the next day, accuracy in preparing the input for each run was critical to progress.

A more resourceful approach to code development was to attempt debug shots in off hours; nights and weekends. These hours were devoted to production runs of the physics codes. The production runs were submitted as decks to the operators by the end of the day, along with fiercely argued priority of when and for how long each was to run. If some of the high priority decks were faulty, or the runs ended sooner than anticipated, the lower priority runs were allowed onto the machine. The tireless code developer, or designer, would spend the nights and weekends making friends with the operator, watching this unfold. And when the "critical" jobs were done, the user would intervene.

From early on, the ground rule was that if a user would go to the effort of working off hours, and acted within reason, the onsite user would have priority over most production runs. A designer working at nighttime could submit a new set of runs, often based on the results of runs the designer had submitted during the day and had now just concluded, assigning time to them from the unused allocation for his or her group's portion of available time machine. Sometimes the designer would even iterate on the results of these before daybreak, thus gaining precious time on the design cycle. A code developer in the night would have spent some time preparing his or her deck, usually a new set of instructions, and then would assist the operator in loading his or her debug shot. The code developer who took the trouble to learn the details of machine operation would offer to do all the operation during the debug shot and even restart the production runs for the operator, thus earning the operator's continued cooperation throughout the night.

The result would be a code available to the designers more quickly than expected or a more robust weapon design. In fact, most of workhorse codes and most of the important designs and design ideas were all developed at least as much off hours as during "normal" hours. Weekends offered much more opportunity for this kind of interaction; sometimes much of a Sunday and the following early morning Monday would be available to the intrepid. To this day, old time computer users, designers and code developers alike, wince when someone says, "have a nice weekend." Weekends aren't to be wasted, they remember, weekends – and especially holidays – are when the real work gets done.

For code developers, the desired outcome of repeated machine interactions – normal or off hours – was clear: a debugged code, modeling the required physics, and running as fast as possible. For the designers, exactly what to extract from the information contained in a completed run required some thought. The end product of a run was usually not a single or summary suite of numbers. Even the calculation of the yield of a design often offered information of explanatory importance because, whatever physical processes were being modeled in a design code, the consequences of the initial conditions and the properties of the materials chosen could be seen unfolding in time.

The purpose of many design code runs was to follow the consequences in time of a sudden burst of energy – the explosion of high explosive in an atomic bomb or the explosion of an atomic device in a hydrogen bomb. Although this burst of energy and its impact was over in a small fraction of a second, this presented no real difficulties to its calculation provided that certain constraints on the time step of the run were followed. Such effects as the ebb and flow of energy through the problem, the acceleration and deceleration of materials or the effect of criticality rising and falling in various areas could be studied through the course of a typical run. For example, if an inner spherical shell was not compressed as much as expected, the reasons why might be found by following the details of the motion of the shells outside of it. Of course the believability of such a calculation was limited by its resolution – the number of zones and time steps, the sufficiency and precision of the physical processes included in the code, and the accuracy of the properties of the materials involved. Studying the time history of the problem also was a way of finding code bugs – a material whose temperature went negative when shocked was likely not of this world.

Since designers did not always know what results of a run would be of greatest interest, they usually elected to have printed as much of it as was practical; the meaning of "practical" being set by the available printing technology. Starting with the 701, IBM printer speeds of about 150 lines a minute were available, each line 120 characters in width. The output of most design codes was formatted to fit 10 columns of 8 to 10 digit numbers plus their exponents into that width. For each snapshot in time selected, the page would start with the problem time and some summary information, followed by lines of numbers, each line representing a zone in the problem. The numbers in the 10 columns for each line would be zone quantities like temperature, velocity, position, or pressure, with perhaps blank lines separating regions of different materials.

During a run the information to be printed would be written to an output tape, and after several runs had filled the tape, it would be moved to a dedicated printer. A typical run might have 50 zones and run for one half an hour, with the number of time snapshots to be printed being an option set in the input deck by the designer. If 100 such printouts were sent to the output tape, it would then take about a half hour to print out the results. At this rate, the offline printer would just barely keep up with the production output of one machine. But as the stable of 704s grew, and as their memory increased from 4000 words to eventually 32,000, allowing several times the number of zones to be included in a run, the existing printers became swamped.

Partial relief came in 1957[36] with the 600 line per minute impact printer from Remington Rand. Besides being dangerously noisy, it was clearly not the solution – the users scaled their output requirements to capture and print even more of their runs. New technology was deployed in 1959 with the Stromberg Carlson SC5000. First an image of the output was created by a CRT with a mask in the electron gun assisting in forming the characters. By transferring the image onto to continuously fed paper with the newly invented xerography process, the new printer produced an output page every second, six times the speed of the impact printer. Its problem was combustion: the high temperature required to fuse the toner onto the fast moving paper sometimes caused the paper to ignite – and spread the fire to the fresh paper continuing to be fed into the fusing oven until the machine could be shut down.

Faster computers and bigger codes soon made even this machine inadequate, forcing another technology advance in 1964, the Radiation Printer. It printed – "Radiation" was the company's name, not its technology – by using tiny electrical arcs to create high contrast marks on electrosensitive paper. Capable of printing seven pages every second, its output was approximately as fast as it could read the results from the output tape. As paper moved past hundreds of electrical styli, arranged in a line of 120 characters each formed by a 7x9 dot matrix, styli appropriate for the next line of print would instantly be charged and then arc through the paper, burning tiny holes in the surface and exposing the black undercoating. The paper was heavy with a greasy feel, dark and hard to read, difficult to write on, and smelled bad – it continued to outgas for months after it was printed. But until microfiche printing fully flowered, for ten years the Laboratory's code developers and designers filled their offices with Radiation paper printouts of core dumps, listings, and the precious details of advanced nuclear weapon designs.

Graphical display would have been a helpful alternative to paper, but in the late 1950s the necessary technology had yet to be invented. Cathode Ray Tubes (CRTs) were not capable of drawing lines: an arbitrary line had to be drawn dot by dot, by the mainframe computer, a slow and expensive process.[37] Extensive use of graphics would have to wait for the development of high speed, off line graphical displays. Also the technology for automatically exposing film to the image created needed refinement and the software to make it all function properly had yet to be written. Still, much progress was made, both by engineering and software efforts at the Laboratory and in the commercial computer industry. Perhaps because these tools began their entrance in the early 1960s, after the designers were getting their printout needs reasonably well met, their adoption by designers was slow.

But the main reason printouts were the dominant record of design is simply the multiple uses they served. Comparing detailed results of the same geometry on two codes, checking with a hand calculation that a new code is producing precisely what it was intended to produce, quickly comparing a series of runs for evidence a new effect, were all actions that could be done with printouts in the office. Better this than searching through old tapes and waiting to go back on a machine to generate graphical evidence at the desired resolution. All the detail needed was at hand, on paper, immediate. It wasn't until decades later such interactions as query, display, and magnification could be done so quickly, so responsively, by accessing a library of completed runs electronically from one's own office.

In 1958, IBM introduced its next model, the 709. Equipped with 32,000 words of core memory and only percentage points faster than the 704, it nevertheless in many details was a more stable, upgraded design. Fernbach exchanged the Laboratory's first 704 for its first 709 in October of 1958, and did a similar swap for its second 709 the following September.

The 709 was in fact a place holder for IBM's first transistorized machine, the 7090, announced at the end of 1958. Dramatically smaller in size and about six times the speed of the 709, the 7090 marked a revolutionary new stage in computer performance. And this advance came with the great benefit of direct compatibility. The "dusty decks" of programs written for the 709 could simply be read into the new machine unchanged to take full advantage of its enhanced power. As soon as deliveries of the 7090 were under way in early 1960, IBM withdrew the 709. The company continued to deliver the 7090 and its upgrades, such as the 7094, throughout the decade of the 1960s.

If vacuum tubes had made electronic computers possible, transistors made them practical. Needing far less volume, floor space, and electrical power then tubes, as well as greatly enhancing computing reliability, transistors allowed computers to not just shrink and speed up but penetrate new markets as well. New computer companies like Digital Equipment Corporation exploited the transistor's flexibility by first selling low cost switching devices for scientific and engineering applications and then, beginning in 1960, packaging them into a series of computers for similar markets. These new computers, with designations like the PDP-1, -6, and -8, equipped with modest memory size and capability but being only the size of a desk and costing of the order of one percent of the 7090's $3 million, would come to be called mini computers.

Mini computers began appearing throughout the Laboratory. In mainframe computer rooms they took on tasks such as auxiliary storage controllers and drivers of stand alone graphic display systems. By assuming roles that previously only a mainframe could perform, mini computers freed mainframes to execute their unique role of heavy duty number crunching. And soon computing centers supporting small systems emerged elsewhere at the Laboratory: in the Engineering Department, in the administrative side of the Laboratory, and wherever experimental data processing was important, like Test Program and the Chemistry Department. This diversification of computing within the Laboratory troubled those who felt that Fernbach should add control of mini computer acquisition and support to his duties. Fernbach felt otherwise. He wanted to maintain his, and therefore the Laboratory's, full focus on acquiring and supporting big computing; called "our life's blood" by Peter Moulthrop, a senior large weapons designer and a 1960s A Division leader.

But the 7090 was not the first transistorized large scale computer to be delivered to the Laboratory. In early 1955, more than five years earlier, after announcing a "strong interest"[38] in high speed transistorized computing, Livermore was visited by executives from both IBM and Remington Rand to discuss options. A year before the first 704 was delivered, IBM was beginning to plan for a bold step in computing design: achieving a 100-fold improvement in performance over the 704, and in April, 1955, proposed such a machine to Livermore. However, the Laboratory, experienced as it was with Remington Rand's Univac, favored their proposal for a major new, transistorized machine. In September, 1955 the Laboratory signed a contract with Remington Rand for delivery in three years, 1958, of the $3 million LARC, the Livermore Advanced Research Computer. IBM refined its proposal and presented it to Los Alamos in late 1955. After extensive discussions, in November, 1956, Los Alamos signed with IBM for delivery in May, 1960 of its advanced computer, the 7030, nicknamed the Stretch.

Like the Univac, and unlike most other computers which presented the programmer binary data and instructions, the LARC was a decimal machine; a feature many welcomed but some felt limited its broader applicability, even in the field of scientific computing. With a 12 digit word length, the machine would offer both single and double precision – single or double word accuracy – floating point hardware. Not fully transistorized, its tube driven core memory systems would provide 10,000 words a cabinet, with as many as ten cabinets being addressable. The cost of memory, and its large size, kept the final version delivered to Livermore at three cabinets, or 30,000 words. A significant augmentation of memory was offered in the form of rotating drum memories, each drum holding 250,000 words. Livermore's twelve large drums were housed in a separate "clean" room needed to protect the delicate drums from contamination. The room's large picture windows showed off the red glow of the drum's spinning surfaces to theatrical effect. The machine's large volume required the construction of a 7,500 square foot building to house the LARC. Both computer support mathematicians from T Division and the Univac engineering maintenance team participated in developing the detailed specifications for the machine. Ed Lafranchi of the engineering team remembers noticing how extensive Livermore's input and output requirements were growing to be and wondering if it would even include voice input.

Remington Rand, in an attempt to inject fresh ideas into the project, hired several recent young mathematics graduates to join the design team. With virtually no schools offering courses in computers at any level, the recruits had no training in computer design and came with a clean slate. Mary Lou Moore from Chestnut Hill College in Philadelphia was one such hire. She and another newcomer were assigned to design the central processing unit, or CPU, the heart of the machine. As she remembers, "The recruiters said nobody has any experience; they just find out if the applicants can think. It's scary when you think about it. They sent two girls who didn't know anything about how to design this thing."[39]

But Mary Lou Moore took to it quickly. "I think the essence of design was to break things down into 'yes' or 'no' – the light was either on or off. It was making things simpler not more complex. And then you chose to take those simple things and join them up. It was like being paid to work a puzzle, and we got to do it all. In the morning, ten people in the design department at Remington Rand would meet to discuss the status of our work, so everyone was clued in. We talked to the Stretch designers at a convention and they had 55 people. And that's harder, because you can't be so sure that everyone is up to speed."

Jim Moore, a Livermore Univac engineer, was named lead LARC engineer and spent months in Philadelphia working with the design team learning the details of the machine that was to be his responsibility when it shipped to Livermore. It sheer size – 80,000 transistors, 180,000 diodes, several hundred thousand wires – made mastering its intricacies a difficult task. Moore remembers his supervisor telling him, "Make sure you bring one of the designers back with you." Moore did so in an unexpected way – Mary Lou Moore became his wife before the LARC was operational in Livermore.

The Moores settled into suburban life with Mary Lou the homemaker while Jim spent his days and some nights helping the LARC adapt to Livermore. On one such occasion, working late after several fruitless hours pursuing why the machine had earlier ground to a halt, Jim received a call from Mary Lou, asking, "When are you coming home to dinner?" After he briefly described what the team was faced with, she said, "Oh, it's probably the ASTZ." Jim recalls, "The next place we looked was the ASTZ signal, and sure enough, that was the problem.… I almost didn't want it to be that…. [With] very little description of the problem, she had hit it on the head."

Manufacturing difficulties delayed delivery of the LARC for 27 months to June, 1960 – one month before the arrival of the first 7090. The LARC's performance, however, was as promised: it met or exceeded all of its specifications. Capable of 125,000 floating point multiplies per second, the LARC had become the world's most powerful computer. It was to hold the title for less than a year.

Despite its capability, the LARC was never to play its anticipated central role in weapons computing. Several factors combined to prevent this from occurring. As the machine's delivery delays increased, reliable code development planning for it became increasingly difficult. Fortran's success on the 704 and 709 in the late 1950s meant most code developers wanted Fortran on new machines. The impending delivery of the compatible 7090 would bring an effortless surge in computing capability to existing codes on the 709 and with two more 7090s in the pipeline, code developers were reluctant to return to the costly process of assembly language coding for an uncertain machine. Finally, internal difficulties meant negotiations with outside contractors for development of a Fortran for the LARC did not get underway until 1960.

Nevertheless, some software efforts were quite successful on the LARC. In particular, the resourceful programming skills of Leo Collins, under the direction of Bill Schultz, brought the Alpha code to the LARC. Having helped develop Alpha on the 701, Collins moving it to the LARC continued Alpha's distinction of being among the first codes ready for a new computer. When the Soviets broke the moratorium in late 1961, Alpha on the LARC provided the needed boost in design capability to prepare for the United States answering nuclear test series.

Another LARC success came from a different quarter. After spending time in Berkeley to earn his Ph.D. in 1957, and after writing another large scale weapons explosion code, Chuck Leith decided to concentrate on a project he had been considering for some time: modeling the weather. He had become familiar with the work von Neumann was doing on numerical weather prediction while working on the Los Alamos measurement analysis with Herb York in 1950. von Neumann and others had been furthering Richardson's early numerical weather work for some time, in recognition that, as Leith remembers, "the big and complicated nonlinear problems which we were facing both in the weapons business and in the weather business could be tackled with the machine he was developing at the time".[40] Leith talked again in the late 1950s with people in the atmospheric modeling business "who encouraged me to get into it even though I had no experience in that area except for a background in mathematics and physics." After receiving approval from the Laboratory for the venture, he worked for about a year on a model on the Univac in preparation for the LARC and spent the summer of 1960 at the International Institute for Meteorology in Stockholm "having access to a library and people who had some familiarity with the nature of the problems encountered in doing this sort of thing."

Shortly after the LARC's arrival, Leith had his weather model up and running. The LARC "was a machine that if one programmed properly could be well balanced in its input and output versus its arithmetic capabilities. If you did so, you could maintain minimum latency – minimum waiting time – on moving data to and from the drums."

Exploiting the drums' large capacity, Leith's code was able to include all the energy, moisture, and wind data necessary to define the atmosphere with 6 layers vertically and each layer defined by zones 5 degrees on both longitudinal and latitudinal sides, with a longitudinal coarsening towards the pole. The time step for each cycle of the calculation was set at 10 minutes. In contrast, Richardson's hand calculation had used one layer of zones – altitude data being almost non existent – with the similar 1.8 degree latitudinal and about 2.7 degrees longitudinal sides but with the much larger, and too large, single time step of 6 hours.

Leith's model initially included only the northern hemisphere of the globe with a reflecting boundary condition placed at the equator. Leith used a novel scheme to initiate the calculation. Working earlier with the 709, "I acquired Teletype information from all the weather stations around the United States and set up an automatic, purely numerical scheme for interpolating between them to generate the atmosphere's initial state." This was later developed further by others with the startup of the numerical weather prediction facility of the National Weather Service. Leith, working closely with the Laboratory's engineering team, also harnessed the LARC's graphical devices to make the first colored movies of numerically predicted global weather patterns. Crude by today's standards, the movies showed dramatically the basic weather patterns of the northern hemisphere: the march of storms down the western coast of the United States and the complex effects of the Gulf of Mexico and Southern latitudes on the flow of low pressure systems northeasterly to Europe. The numerical results even showed evidence for a seven-day weather cycle. Using an electronic computer, Leith had succeeded in building a Richardson "Forecast Factory".

Several others at Livermore also began making pioneering contributions to science using the Laboratory's high speed computers. In 1962 Sterling Colgate and Richard White of T Division published their results of the first calculation of a supernova, which predicted, later experimentally confirmed, the large flux of neutrinos accompanying the explosion. After Colgate left to head up the New Mexico Institute of Mining and Technology, White teamed with Michael May to extend the calculation by including, for the first time, Einstein's general relativity to probe the mystery of a supernova's remnant. Also, at roughly the same time, James R. Wilson, originally of T Division, later of B Division, began an ongoing, 40-year effort in computational astrophysics while also creating advanced design codes for the Laboratory's weapons program. Recognized around the world as a founder of computational astrophysics, and the winner of the American Physical Society's 2007 prestigious Hans Bethe Prize, he and his successful students have forged significant advances in the field.[41]

The month LARC was being installed in Livermore, June, 1960, IBM ran ads in major newspapers for its new 7030, the Stretch computer.[42] Several orders had been placed, chief among them an order from the National Security Agency for an expanded version of the machine to be called Harvest. With serial 1 originally scheduled for delivery to Los Alamos in this same period, the machine's production was running about a year late. More disturbing were the muted speed comparisons made by the Stretch team at the Eastern Joint Computing Conference a few months before. Nevertheless intrigued with the 7030's potential, at the end of 1960 Lawrence Livermore signed a $13.5 million contract with IBM for delivery of a 7030. The fully transistorized machine featured 96,000 words,[43] three times the words of any other computer at Livermore, and was still said to be meeting its original performance objectives. Just in case it turned out otherwise, some very specific speed requirements were agreed to and called out in the contract. When careful measurements run a few months later by both IBM and Livermore showed the performance of the 7030 would in fact be only half of the promised values, Thomas J. Watson, Jr. announced that the price of the machine would be cut in half, that those seven orders already placed would be honored, and that the 7030 would be withdrawn from the market. The Los Alamos 7030 was delivered in April of 1961 and its performance, though marked a failure, was still good enough – almost a factor of two faster – to wrest the Top Computer crown from the LARC. Livermore's 7030, serial 2, was delivered at the end of 1961.

Although the usage of the 7030, like the LARC, started out slowly, it gradually became less of a niche machine than did the LARC. With its high speed and large, fast core memory, the Stretch attracted codes that handled very large amounts of data and were used to make exceptionally long running calculations. For instance, Harry Nelson programmed the Alpha code on the Stretch to exploit its advanced capabilities. Design teams would use the smaller 7090s to do a series of fairly short running parametric study of a design and then, selecting a configuration and conditions that seem promising, set in motion on the Stretch an in depth calculation with those specifications that might run several hours. As more graphical features were added to the 7090s, the linkage between them and the Stretch became even stronger, with the Stretch output tapes being carried to the 7090 and graphical records made from them. Even real time linkages were exploited. During an intense design cycle of a major weapons system in the late 1960s, the design team developed for the 7090 the capability of recasting the grid from a large Stretch design code. When the Stretch calculation would run into difficulty with its grid becoming scrambled, the designers would transfer the crippled calculation to tape, read it into the 7090, and using a light pen featured with the DD80 graphical display system connected to the 7090, quickly and accurately recast the grid to a more calculable form, conserving mass and energy in the process. This result would then be carried back to the Stretch and the calculation resumed. Many such productive predawn sessions were necessary to produce a successful design.

The LARC and the Stretch, both the product of Livermore's 1955 request for a great leap forward in scientific computing, shared one final similarity: they represented the last major scientific computers either company would produce for a very long time. The LARC, with its considerable delay and $13 million budget overrun,[44] was limited to just two sales – the other machine being installed at the Navy's David Taylor Model Basin. From then on Remington Rand, by now Sperry Rand, would pursue development of the transistorized Univac series, aiming toward the more lucrative business market. IBM went on to develop the successful System/360 series of computers, also largely intended for the business sector. IBM's early estimation that the 7030 had failed as "A Stretch Too Far" was later moderated. The transistorized core memory system developed for the Stretch was the memory system of the successful 7090 series, and many features introduced by the Stretch project found their way into System/360. These innovations created profits for the company that more than paid back the 7030 deficit. The 1954 co-originator of the Stretch project, Steve Dunwell, later its head, was restored to prominence in 1966 and made an IBM Fellow.[45]

Just as the major computer manufacturers were turning away from building new flagship computers, two independent developments were already underway in the early 1960s that would rescue and reset the future of scientific computing at Livermore. The rescue would mean new machines from a new quarter; the reset would enable a completely new and more productive method of their use.

As for the rescue, Control Data Corporation's brilliant computer architect Seymour Cray was already at work in a field he would dominate for the next 30 years. Cray, born in 1925, was drafted in World War II and served in both the European and Pacific Theaters – the latter as a Japanese code breaker. After graduating from the University of Minnesota in 1950 with an Electrical Engineering degree, he went to work for ERA, the Engineering Research Associates of St. Paul, Minnesota. ERA was headed by William Norris, who had also worked during the war as a cryptographer and had helped form the company to build code breaking machines for the Navy after the war. ERA moved from building code breaking machines to more general purpose machines, but after political and legal difficulties drained the company's resources, ERA was purchased by Remington Rand in 1952. When Sperry bought Remington Rand in 1955, the two computer entities – ERA and Univac – were merged into one, and much of ERA's work was abandoned.[46] Most of the ERA employees left, and led by Norris, formed Control Data in 1957, with Cray joining them a year later.

Cray's first effort for the new company was the 1604, a transistorized machine with the capability of a 7090 that began shipping in 1960. The Laboratory, always on the lookout for new directions in computing, sent the originator of the 1958 first "Fortraned" weapons design code to Minneapolis to check out the machine. Since CDC had already developed a Fortran for the 1604, it was a simple matter to see if it could compile the weapons code's "dusty deck". Within 24 hours, the 1604 was running the program and Livermore was ready to begin its decade long relationship with Control Data. CDC's next machine was the 3600, while Cray went from the 1604 to develop his first super computer, the 6600, released in 1964 as the world's fastest computer. Cray's design of the 6600 balanced processor and memory speed to achieve performance of from two to three million floating point operations per sec, or MFLOPS, for a wide variety of programs. Before the decade of the 1960s was out, Cray's next machine, the 7600, had raised the performance bar by a factor of five. Cray went on to form his own company, Cray Research, and produced machines that earned the company a clear monopoly of state of the art of high performance computing into the 1990s and Cray himself the undying gratitude of generations of computational scientists the world around.

The reset of the use of scientific computers was an undertaking of an entirely different sort. At Livermore, it grew from the discussions Hardy had with others about the notion of treating a computer center not as a series of disconnected machines but as a connected, multi-capable resource, with each machine assigned to tasks for which it was best suited, different machines all addressed by the user through a common interface. To this was added the concept of more than one user using a particular machine at the same time, sharing time with other users, each user potentially gaining access remotely and at will. These ideas together became known as the Octopus system. Its development would take a large portion of the decade and require mini computer invention, super computer adaptation, extensive network construction, considerable software development, and produced the tensions that accompany major readjustments of behavior.

By the 1960s computer manufacturers were offering so-called operating systems for their machines, which the Laboratory would modify and later build independently. The systems would run more or less automatically the series of programs submitted as decks by different users. This "batch mode" usually made more efficient use of the machine than manual operation but it made user interaction with a machine more difficult. Each machine had its own system and except through transferring tapes, no connection with other computers was possible. The Laboratory's multiple computers, along with the even more capable computers on order, placed increasing demands on operating such a large, diverse, and increasingly dispersed center for what was now a clientele of over 1000 users. The notions embodied in the Octopus proposal suggested it would be an attractive solution but such a major innovation required careful research before implementation.

The first opportunity Hardy had to study the time sharing notion came in the summer of 1963 when he spent six weeks at the MIT computer laboratory with an invited industry study group, attending lectures on time sharing ideas and using MIT's newly developed CTSS, Computer Time Sharing System. CTSS ran well on a 7090 configured with Teletype connections to accommodate up to 24 users, allowing users virtually continuous interactivity while sitting at Teletypes in their offices. Although Livermore's center was far more complex than MIT's, Hardy was inspired by this simple demonstration. After some difficult discussions with Fernbach, Hardy initially with the help of small team proceeded to develop a similar system for the 6600, due for delivery in September, 1964. Fernbach's caution stemmed from his perception that the enterprise was still in the research phase and might cause a significant disruption and drain of resources without leading to success – a classic clash of vision with practicality.

By the summer of 1965, parts of a system of 48 Teletypes communicating with the 6600 from remote stations were in fairly good operation, with reliable time sharing service being delivered in 1966.[47] Before the end of the decade Octopus was connected to several mainframe, or "worker", computers and hundreds of users. Among its unique features was its large scale archival storage device, the film-based IBM Photostore, capable of holding in active storage a trillion bits of information. Data was recorded on small film chips by a CRT-like device, chemically developed, and physically filed into small canisters that were then tucked into large trays, trays capable of holding a total of over 140,000 chips, each ready for rapid readout – all this, automatically. Astonishingly, this Rube Goldberg device worked, and worked well, for over two decades, thanks to its adept IBM customer engineer, James Dimmick. In its lifetime of service to Livermore, the Photostore was filled to capacity thirteen times over.[48]

The Laboratory's Octopus system has taken on many forms over the years but it remains today as the way in which users and code developers at the Laboratory get their work done. When asked about Octopus by Fernbach's 1989 interviewer, Marilyn Ghausi, "Whose idea was it? Was it a group?" Fernbach replied, "It was a group, but the main instigator was Norman Hardy."[49]

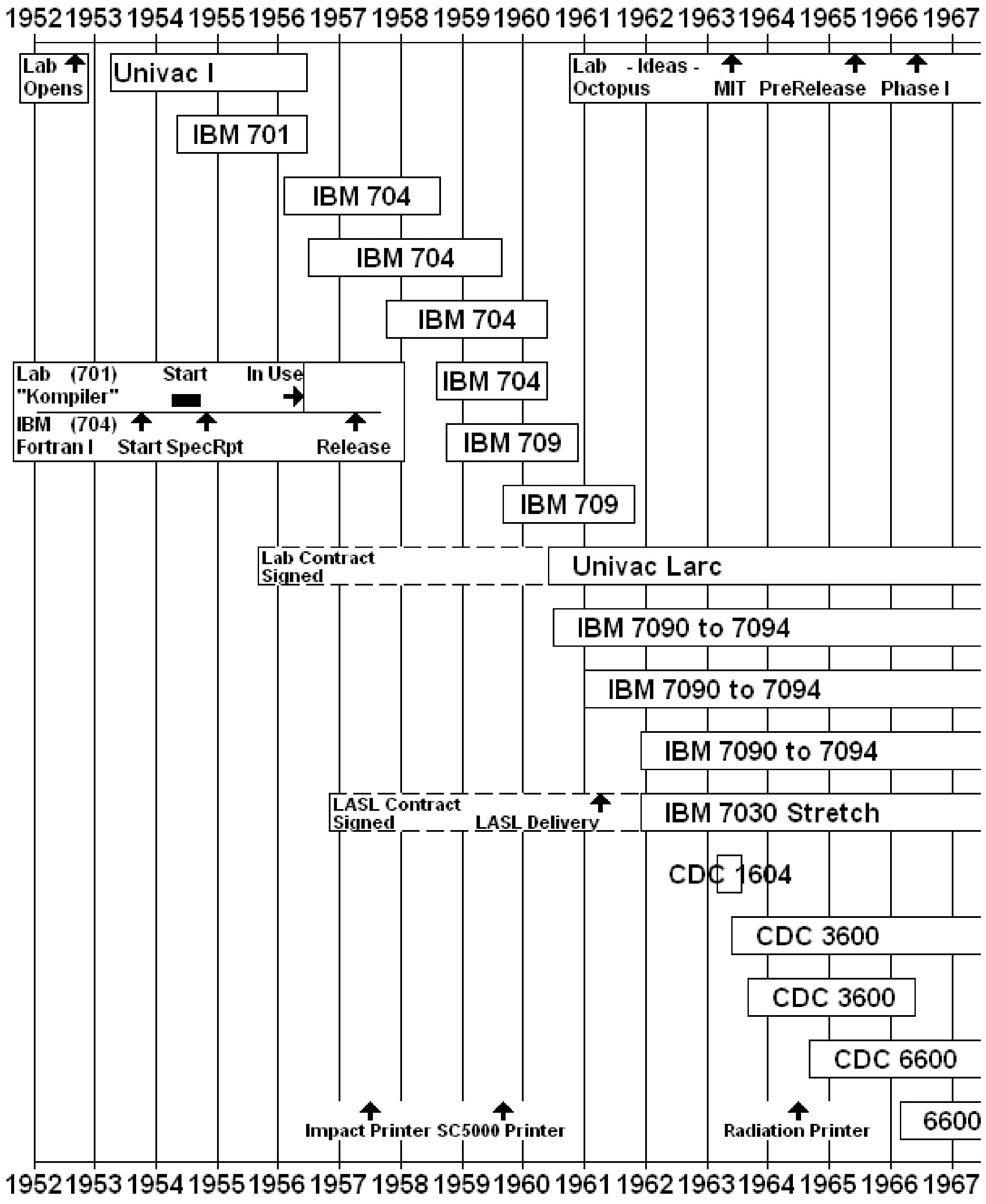
At Livermore, the insights into understanding weapon physics, the world's climate, stellar evolution, and complex biological processes – to name a few – continue to be deepened by the increasing performance of high performance computers. And performance enhancement continues its march ever higher. The millions of numbers a second of the 1960s have today become hundreds of millions of millions, on their way, perhaps, to millions of millions of millions – each second.

> *It is 11 pm on a warm summer evening in 1958 at The Creamery, Livermore's downtown soda shop, as Norman Hardy and his friend arrive for their ice cream soda break from the Laboratory, three miles out on East Avenue. Mid-day break, for the second half of their day often ends at sunrise. Immediately after ordering milkshakes, Norman drops on the counter his extra wide, fan folded computer listing and flips it open, bottom to top. As he banters with his friend on amusing, unrelated matters, Norman swoops through the pages, making quick pencil marks as he goes.*
>
> *The ageing, gentlemanly proprietor serves their order and stands behind the counter, frowning while he studies Norman's pencil darting up and down the listing. Norman is debugging his binary tape reader code, written in the hieroglyphics of the local computer assembly language and annotated with the multi digit values of many memory locations. From the proprietor's position opposite, the listing appears, and is, utterly incomprehensible but, being machine printed on lined paper, worthy of respect. His expression serious, even grave, he is finally moved to caution his young visitor from a disturbing future. He says, "You've got a lot of numbers there, young man."*

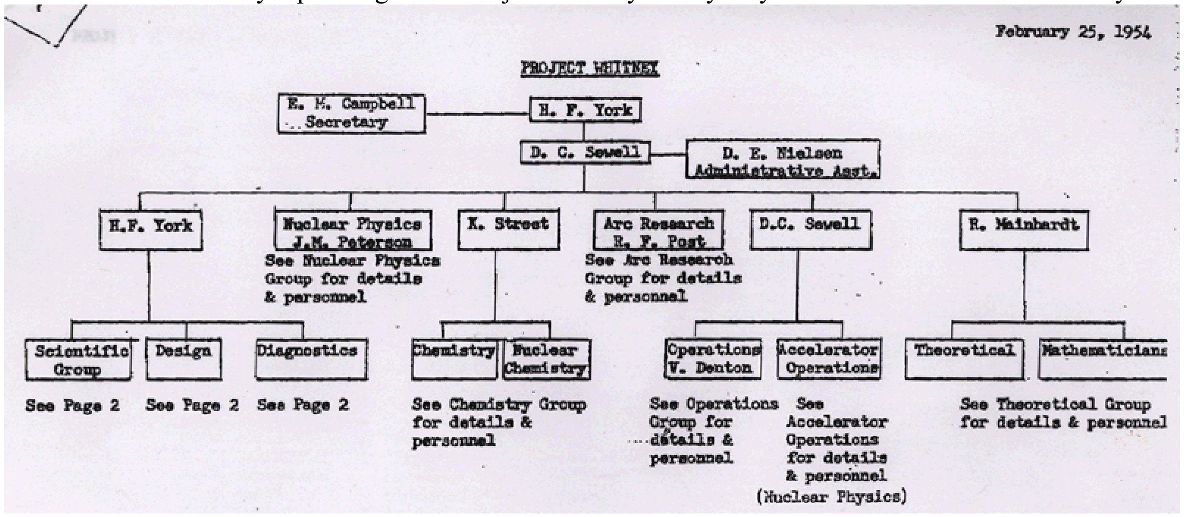
Acknowledgements

The author would like to thank C. Bruce Tarter for suggesting the topic; E. Maxine Trost, LLNL archivist, for vital assistance in artifact recovery from the indispensable Livermore Archives; George Michael for recording extensive verbal histories of early Livermore employees, only some of whom are referenced in this paper; and the suggestions of Paul Chrzanowski and of those who were active in the subject time period: Ann Hardy, Norman Hardy, Roland Herbst, Chuck Leith, and George Maenchen. Unreferenced details are based on the author's personal experience or opinion. Finally, the author accepts full credit for all errors.

Livermore Computer Acquisition Chart



[1] http://www.lanl.gov/history/postwar/debate.shtml
[2] http://www.lbl.gov/Science-Articles/Archive/early-years.html
[3] http://www.cs.yale.edu/homes/tap/Files/ada-lovelace-notes.html
[4] G.H.Hardy, *A Mathematician's Apology*, Cambridge University Press, 1940.
[5] http://aleph0.clarku.edu/~djoyce/hilbert/problems.html
[6] Alan Turing, On computable numbers, with an application to the Entscheidungsproblem, 1936. http://www.emula3.com/docs/OnComputableNumbers.pdf
[7] http://inventors.about.com/library/weekly/aa060298.htm
[8] A retyped copy with an introduction by Michael Godfrey is available at this site: http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf
[9] Alan Turing, ACE Report, 1936. http://www.emula3.com/docs/Turing_Report_on_ACE.pdf
[10] Scott McCartney, *Eniac*, Walker and Company, 1999
[11] GAM-Chuck Leith 1994. (Many of the personal recollections referenced in this paper come from unpublished interviews with early Laboratory employees conducted by Livermore's George A. Michael. They have been recently placed on the web at http://www.computer-history.info/ A reference to one of these interviews will be represented by the prefix GAM, followed by the name of the interviewee and the date of the interview.)
[12] The GAM website http://www.computer-history.info/Page1.dir/pages/Fernbach.html includes Michael's summary and evaluation of Fernbach's career, along with an interview of Fernbach conducted by Marilyn Ghausi of UC Davis in 1989, two years before Fernbach's death.
[13] This chart shows early top management. Project Whitney was synonymous with Livermore Laboratory.


February 25, 1954
PROJECT WHITNEY
E. M. Campbell Secretary
H. F. York
D. C. Sewell
D. E. Nielsen Administrative Asst.
H.F. York
Nuclear Physics J.M. Peterson
See Nuclear Physics Group for details & personnel
X. Street
Arc Research R. F. Post
See Arc Research Group for details & personnel
D.C. Sewell
R. Mainhardt
Scientific Group
Design
Diagnostics
See Page 2
See Page 2
See Page 2
Chemistry
Nuclear Chemistry
See Chemistry Group for details & personnel
Operations V. Denton
Accelerator Operations
See Operations Group for details & personnel
See Accelerator Operations for details & personnel (Nuclear Physics)
Theoretical
Mathematicians
See Theoretical Group for details & personnel

Notice York heads the proto- A, B, and L Divisions, as well as the Laboratory at large. York and his team of then UC Radiation Laboratory diagnosticians, including Harold Brown and Michael May, had successfully measured neutron output from the George event at Eniwetok in April, 1951, over a year before the founding of the Livermore Laboratory. This chart and its further details are available in the Livermore Archives.
[14] Lewis F. Richardson, *Weather Prediction By Numerical Process*, Cambridge University Press, 1922; Dover Press 1965.
[15] http://www.spsnational.org/ford_article.htm
[16] GAM-Cecilia Larsen 1996
[17] GAM-Ed Lafranchi 1997
[18] GAM-Chuck Leith 1994
[19] http://www-03.ibm.com/ibm/history/exhibits/701/701_feeds.html and http://en.wikipedia.org/wiki/UNIVAC_I
[20] http://ed-thelen.org/comp-hist/TheCompMusRep/TCMR-V07.html
[21] http://www.idi.ntnu.no/emner/dif8916/denning.pdf
[22] Author interview with Roland Herbst, 2006

[23] GAM-Norman Hardy 1994
[24] John Backus, The History of Fortran I, II, and III, *IEEE Annuals of the History of Computing* v20n4: 68-78, 1998.
[25] A brief comparison of Kompiler 2's syntax with several other early compiler languages can be found in this superb review paper: Donald E. Knuth and Luis Trabb Pardo, The Early Development of Programming Languages, (reprinted in) *A History of Computing in the Twentieth Century,* 197-273, Edited by N. Metropolis, J. Howlett, and Gian-Carlo Rota, Academic Press, 1980. (Originally printed in J. Belzer, A. G. Holzman, and A. Kent (eds.), *Encyclopedia of Computer Science and Technology,* V6: 419-493, Decker, New York, 1977.) In a summary "rating" chart of features and impact of world history's first 20 compilers, Knuth & Trabb give Kompiler 2 ratings of C's in most features but F in impact. Given its role in the timely creation of the Alpha code and Herbst's weapon design, firmly establishing Livermore credibility in thermonuclear design, we would argue Kompiler, Elsworth, and his group deserve an A. The programmers' manual for Kompiler 2: "Manual for Kompiler 2", by Elsworth, Kuhn, Schloss, and Tiede, November 7, 1955, is available in the LLNL Archives.
[26] http://cap-lore.com/Languages/Kompiler.html
[27] http://www-03.ibm.com/ibm/history/exhibits/mainframe/mainframe_PP704.html
[28] http://en.wikipedia.org/wiki/Core_memory
[29] Steve Lohr, *Go To*, Basic Books, 2001.
[30] Several features of Fortran changed during its development. Fortran's original syntax, sometimes referred to as Fortran 0, is described in this unpublished IBM document. For instance, user defined variable names were initially allowed to be only one or two characters in length. Other differences include an early version of the DO statement, no FORMAT statement, different syntax for the IF statement, and no comment cards. A copy of this headwater prospectus is available in the LLNL Archives.
[31] GAM-Robert Hughes 1997
[32] http://www.fortran.com/FortranForTheIBM704.pdf
[33] http://www.acm.org/classics/oct95/
[34] GAM-Sam Mendocino 1996
[35] http://en.wikipedia.org/wiki/IBM_704
[36] George Michael's detailed and amusing account of the development of printing at Livermore is recommended reading at http://www.computerhistory.org/archive/core_1_4.pdf#search=%22george%20radiation%20printer%20livermore%22
[37] Supposedly plotting at the rate of 8,000 dots per second, the IBM 740 CRT recorder, featured on both the 704 and 709, would take as much as a second to draw several lines. Making a full movie of the grid moving in an explosion, giving the designer a gestalt view almost impossible to glean from printouts alone, was seldom done: it took coordination, perseverance, and required hours of coveted machine time.
[38] http://archive.computerhistory.org/resources/text/IBM/Stretch/102636400.txt
[39] GAM-Jim and Mary Lou Moore 2001
[40] GAM-Chuck Leith 1994
[41] For example: James Wilson, *Relativistic Numerical Hydrodynamics,* Cambridge University Press, 2003.
[42] http://archive.computerhistory.org/resources/text/IBM/Stretch/102636400.txt
[43] Since each word on the Stretch was 64 bits in length, much longer than the IBM 700 series 36 bit words, the Stretch memory's bit capacity was in fact nearly six times that of the memory of other computers at Livermore.
[44] Gray, G.T.; Smith, R.Q., Sperry Rand's First Generation Computers, 1955-1960, *IEEE Annuals of the History of Computing* v27n4: 20-34, 2004
[45] http://ed-thelen.org/comp-hist/vs-ibm-stretch.html
[46] http://en.wikipedia.org/wiki/Engineering_Research_Associates
[47] Sid Fernbach, *The LRL Octopus System* Internal LLNL Memo, 1968. Copy available in LLNL Archives.
[48] http://en.wikipedia.org/wiki/IBM_1360
[49] The Ghausi interview at the GAM website http://www.computer-history.info/Page1.dir/pages/Fernbach.html