Appendix A

A Short Primer on Using UTEC

This appendix serves two purposes. The first is to provide a more thorough explanation of the design and operation of the UTEC Mark I prototype than was presented in chapter 2. The second purpose is to explore the only UTEC code known to exist, and provide a transcription of selected examples. As the only surviving code written in Canada by Canadians for the first Canadian electronic digital computer it deserves preservation, and in doing so bestows light on how UTEC could have been used for practical computation, limited as it was.

A.1 Design and Operation of UTEC

After the UTEC project ended and the final demonstration was held at the ACM meeting in September 1952, the remnants of the machine were eventually dismantled and it is believed that any valuable parts were cannibalized or sold for scrap. No physical component of the UTEC project larger than a vacuum tube is known to have survived. Extant documentation is also limited. Few formal design documents or blueprints were preserved by University of Toronto staff, and many of those held by private individuals were lost to the normal ravages of over-exuberant housecleaning. The situation is not helped by the fact that there has been no contact by historians or archivists with Ratz since he left the project and academia in 1952. Within the University of Toronto Archives, there exists a single relevant box of material collected primarily by Gotlieb. The box contains progress reports, the occasional diagram and plans, but no blueprints.[1]

That said, there are several publications with comprehensive descriptions of UTEC. The first can be found in the proceedings of the 1952 ACM meeting. The meeting delegates were given a demonstration of UTEC, and R.F. Johnston, who was primarily responsible for the input and output design and implementation, wrote an article summarizing the purpose, operation, circuitry, and use of UTEC.[2] The other important text to depict UTEC is Gotlieb and Hume's 1958 *High-Speed Data Processing*. Following several chapters that explain the functional units of a modern computer, the authors select UTEC as an example of an"extremely primitive machine" that can be examined as a whole, a discussion that occupies only five pages.[3] While these two sources provide details concerning the design and operation of UTEC, a third offers an intriguing look into what it was like to write programs for UTEC. B.H. Worsley, a staff member of the Computation Centre throughout its entire existence, wrote the bulk of her doctoral dissertation at Cambridge, but completed writing in 1952 in Toronto.[4] It included a comparison of the EDSAC, the Manchester Mark I, and UTEC, though she was less concerned with the hardware than the preparation of programs for each machine. Her program examples will be presented in a later section. Unfortunately, although it is known that several UTEC reports describing multiword arithmetic subroutines were written, they cannot be located. The reports were not published but had limited circulation internally among the UTEC team members. Fortunately, the two people most responsible for the design and construction of UTEC, J. Kates and A.G. Ratz, also wrote doctoral dissertations related to the computer research under-

[1] UTARMS B1988-0069, Box 01.

[2] Johnston, "The University of Toronto Electronic Computer", 154–160.

[3] Gotlieb and Hume, *High-Speed Data Processing*, 67.

[4] Worsley, "Serial Programming for Real and Idealized Digital Calculating Machines".

taken during the project. Kates' subject matter – his own theory of the operation of Williams tubes – is only peripherally related to the operation of UTEC as the particular tubes were not used on the prototype, but Ratz's dissertation provides an important glimpse into his thinking regarding the design of UTEC, and the arithmetic unit in particular.[5]

Unfortunately, recreating a complete understanding of UTEC is impaired by the prototype nature of the machine. Some specifications were relatively inflexible, but others changed frequently as various ideas and hardware components were tested. For example, the design of the parallel primary store implied that 12 bit word size was more or less invariant, but there were difficulties with the actual implementation of the Williams tubes. Although the plans called for a 512 word store, only 256 words (the even numbered storage locations) were available most of the time. More important, the instruction set was changed several times to accommodate lessons learned in the testing phase. Other modifications and improvements, less critical to the overall logical design but important nonetheless were frequent and common, as befitting an experimental prototype. Power supplies were changed, different vacuum tubes and storage tubes were used at different times, and an attempt was made in mid-1952 to add a magnetic tape system for auxiliary storage. Thus there is no singular and definitive description of UTEC, but by combining the various sources it is possible to arrive at an approximation.

UTEC was a parallel, binary, one-address digital computer.[6] Physically, it had about 800 vacuum tubes, and stood approximately six feet high, eight feet wide, and one foot deep. Twelve Williams tubes operated in parallel to provide 512 words of storage. UTEC used a 12 bit word, of which 3 bits specified one of eight instructions

[5] Kates, "Space Charge Effects in Cathode-Ray Storage Tubes" and Ratz, "The Design of the Arithmetic Unit of an Electronic Digital Computer".

[6] For an introduction to the principles of computer architecture, the reader is encouraged to consult the early chapters of Gotlieb and Hume, *High-Speed Data Processing*, or for a more historically nuanced approach, Ceruzzi, *A history of modern computing*, 58–64.

and 9 bits pointed to the address – when the store was not working properly and only 256 bits were available, 8 bits were used to indicate the storage location. Otherwise, the 12 bit word referred to a signed 11 bit number, or about 3 decimal digits.

Input and output on the completed prototype was handled with modified six-hole Flexowriter paper tape equipment. In general, only four of the six holes held significant data. Both octal and binary coded decimal (BCD) numbers could be used to represent a word. In the former case, four rows of octal were used – one row for the instruction and three for the location – while in the latter case, three rows of BCD represented a single three digit number. Special symbols such as Stop Input or Decimal Input did use all six holes.[7] A switch on the control panel toggled between octal and decimal output.

The basic operation of UTEC depended on a handful of registers and counters: the 12 bit accumulator and 12 bit arithmetic register for arithmetic, the 9 bit control counter – similar to a program counter in today's terminology – which pointed to the storage location of the next instruction, the control register which held the instruction as it was decoded, and the 12 bit storage register for temporary storage during transfers to and from the accumulator, store, and input-output. The basic machine cycle was split into four periods, named A, B, C, and D. Each period required 30 microseconds and so a full clock cycle was 120 microseconds long. During A time, the control counter is read to determine the location of the next instruction. In period B, the instruction itself is read from the store into the control register and the control counter is incremented to point to the next instruction. In the C time, the instruction in the control register is decoded and in D time the instruction is carried out. To provide program branching, some instructions modified the control counter to point to a specific storage location and instruction rather than one sequential to the last.

As work on UTEC progressed, several different instruction sets were used. Table

[7] Johnston, "The University of Toronto Electronic Computer", Figure 9.

A.1 contains a full set of the twelve different instructions proposed or implemented; as a maximum of eight could be used at any one time, the last five columns indicate approximately when each instruction was in use. The initial set was described in a September 1951 progress report (column 1951).[8] The set was not considered complete or optimal in anyway, but intended instead for testing purposes. In particular, it was expected that instructions T and U (transfer accumulator contents and unconditional transfer of control) would be replaced at a later time by more useful ones. By way of comparison, the Manchester SSEM initially had only seven instructions in June of 1948.[9] Though it and the 1951 UTEC set were similar in purpose, there were a few significant differences. The most obvious is that SSEM lacked input and output instructions, in all likelihood because at that point the Manchester prototype lacked input and output devices accessible to a programmer. Instead, this activity was accomplished by hand and visually. Difficulties getting the UTEC input and output tape system working may help explain why this practice was also adopted in Toronto and why both instructions were subsequently dropped until the end of 1952.

Worsley's dissertation includes a snapshot of the instruction set (column 1952a).[10] Though she finished writing around May 1952, her version of the instruction set and account of UTEC's features indicate that her familiarity with the machine began sometime prior to March 1952. At that time, "several relatively minor modifications were undertaken . . . to make the Model a more satisfactory computing instrument" including "a more useful set of orders" (column 1952b).[11] The changes to the UTEC instruction set since September 1951, including those described by Worsley in March 1952, are directly related to increased experience using the machine and writing programs.

[8] Computation Centre Progress Report, October 1, 1950 to September 30, 1951, UTARMS B1988–0069, Box 1, Folder 2.

[9] Napper, "The Manchester Mark 1 Computers", 367.

[10] Worsley, "Serial Programming for Real and Idealized Digital Calculating Machines", 44.

[11] Computation Centre Progress Report, 1 January 1952 to 31 March 1952, UTARMS A1968–0007, Box 110, Folder 4. Worth noting is that these modifications took place after the decision was made to acquire the Ferranti Mark I and cancel the full-scale UTEC plans.

The input and output instructions were removed temporarily as these could be performed manually at the console until the tape reader and writer were operating more consistently. Until then, an output instruction was reserved and a halt instruction was added. More important was the introduction of the K instruction, which transferred the contents of the accumulator to storage without the sign bit, which was instead put in the least significant position of the accumulator. Though it was not functioning for Worsley, it was in effect, a carry mechanism to facilitate multiple word arithmetic. With UTEC's 12 bit word and small store, this was essential for non-trivial computations as it "materially shortens most routines."[12] Worsley's notes make clear that multiple word arithmetic subroutines had been written by this time, though the details and code do not seem to have survived. By September 1952 (column 1952c) and the ACM meeting, the input and output instructions were operating.

Gotlieb and Hume provide the final UTEC instruction set in their book (column 1958), though they warn that "different combinations of instructions were tried to see how the programming was affected, but for the purposes of this description the instruction code [below] will be assumed."[13] There is one large difference between this set and all 1952 sets: the removal of the K instruction and the gain of the R instruction, which shifted the contents of the accumulator to the right one bit. In any number notation, a right shift is the same as dividing by the base; in this case it divided the accumulator by 2. As UTEC lacked a multiplication or division instruction, the R instruction would have dramatically improved the speed of those programs that could take advantage of it. However, it is not clear if the instruction was implemented for UTEC, or if it is merely an example intended for the book. A block diagram included by Gotlieb and Hume shows clearly how it could have been supplied, but unless changes were made to the adder hardware, the K instruction would have been substantially more useful.

[12] Johnston, "The University of Toronto Electronic Computer", 154.
[13] Gotlieb and Hume, *High-Speed Data Processing*, 68.

Table A.1: UTEC Instruction Sets

Instr.		Description	1951	1952a	1952b	1952c	1958
A	s	Add the contents of s into the accumulator	$\star$	$\star$	$\star$	$\star$	$\star$
S	s	Subtract the contents of s from the accumulator	$\star$	$\star$	$\star$	$\star$	$\star$
T	s	Transfer the contents of the accumulator to s	$\star$	$\star$			
t	s	Transfer the contents of the accumulator to s and clear the accumulator to zero	$\star$	$\star$	$\star$	$\star$	$\star$
R		Shift the contents of the accumulator right one bit					$\star$
K	s	Transfer $0a_1a_2 \ldots\ldots a_{11}$ to s and clear accumulator to $00 \ldots\ldots 00a_0$		$(\star)$	$\star$	$\star$	
C+	s	Conditionally transfer control to s if the contents of the accumulator ≥ 0		$(\star)$	$\star$	$\star$	$\star$
C-	s	Conditionally transfer control to s if the contents of the accumulator < 0	$\star$	$\star$			
U	s	Unconditionally transfer control to s	$\star$	$\star$	$\star$	$\star$	
I	s	Input one word from the paper tape to s	$\star$			$\star$	$\star$
O	s	Output the contents of s onto paper tape	$\star$		$\star$	$\star$	$\star$
H		Halt			$\star$		$\star$

Notes: – s is a storage location
– the 12 bits of the accumulator are $a_0a_1a_2 \ldots\ldots a_{11}$
– $(\star)$ indicates an instruction that was not working at the time.

A.2 Programming UTEC

In various UTEC reports, references were made to routines that computed e^{-x}, $\sqrt{x}$, and even the elliptic integral defined by:

$$K = \int_0^{\Pi/2} \frac{d\Phi}{\sqrt{1 - msin^2\Phi}}$$

but code for these examples cannot be located. Furthermore, it is known that subroutines to handle multiple word arithmetic were conceived of at least as early as 1950, and eventually implemented during the testing phase of the machine. Although the routines did not survive (or cannot be located), a timing table for mathematical operations employing one to four words has been reproduced as table A.2.[14] Thus UTEC could compute with accuracy as high as 13 decimal digits, which was comparable to

[14] Johnston, "The University of Toronto Electronic Computer", 160.

Ferut, but under those conditions was considerably slower than the British computer. Yet on a more level playing field, if a program did not resort to multiple word subroutines, UTEC was faster thanks to its parallel architecture.

Table A.2: UTEC Multiple word mathematical operations

	Word Extent							
	1		2		3		4	
Operation	n	Time	n	Time	n	Time	n	Time
Addition	2	240 μsec.	6	720 μsec.	9	1.1 msec.	12	1.4 msec.
Subtraction	2	240 μsec.	8	950 μsec.	15	1.6 msec.	18	2.2 msec.
Multiplication	2	18 msec.	42	70 msec.	56	120 msec.	70	260 msec.
Division	34	36 msec.	58	120 msec.	80	300 msec.	100	500 msec.
Complement	1	120 μsec.	6	720 μsec.	10	1.2 msec.	14	1.7 msec.
Modulus	4	480 μsec.	10	1.2 msec.	14	1.7 msec.	18	2.2 msec.
Square Root	36	200 msec.	57	2 sec.	80	5 sec.	100	14 sec.
Zero Test	6	720 μsec.	10	1.2 msec.	14	1.7 msec.	18	2.2 msec.
Input Decimal		400 msec.		800 msec.		1.2 sec.		1.6 sec.
Input Conversion	120	100 msec.	150	140 msec.	170	200 msec.	215	250 msec.
Output Decimal		400 msec.		800 msec.		1.2 sec.		1.5 sec.
Output Conversion	75	16 msec.	100	45 msec.	125	120 msec.	150	250 msec.

n refers to the number of instructions.

A.3 B.H. Worsley's UTEC Code

B.H. Worsley was one of the first two employees of the Computation Centre (the other was J.P. Stanley), hired in January 1948 by B.A. Griffith to operate the IBM 602A he had recently rented. Both attended training sessions organized by IBM on the operation of the 602A and were involved in the use of the punched card calculator at Toronto. In the fall of that year, Worsley travelled to the University of Cambridge to study the ongoing EDSAC project as work began on UTEC. Stanley followed her to Cambridge shortly later. Neither the NRC or DRB had approved of these junkets, but both were permitted to stay until EDSAC was operational the following spring. Not long after, Stanley returned with a table he had computed on EDSAC, but Worsley remained, now registered as a Ph.D. student at Newnham College of the University of Cambridge, supervised by D.R. Hartree.[15]

Her dissertation, "Serial Programming for Real and Idealized Calculating Machines", presented several numerical solutions to scientific problems (which she had solved with EDSAC) but was preoccupied by the notion of an "optimum basic universal digital machine."[16] That is, she hoped to discover a general set of computer instructions that could be implemented universally and optimally. As part of her study, she compared the three physical computers with which she was most familiar: the EDSAC, the Manchester Mark I, and UTEC. A description of each machine was provided, including physical characteristics, the preparation of programs, and noteworthy idiosyncrasies. This was followed in the first appendix by several programs written for all three machines, some trivial in nature but others significantly more complex.

A selection of her UTEC programs will now be presented, using instruction set 1952a. First, a short review of her notation:

a	the octal number 'a'
(a)	the storage location 'a'
C(a)	the contents of storage location 'a'
Acc.	the accumulator

At the time she wrote these programs, the odd numbered storage locations were not available, nor were the K and C+ instructions. Instead, the C- instruction was used, being nearly equivalent to the latter, and the K instruction is irrelevant to her examples.

A standard convention using four columns was used to transcribe the programs to paper. The first column from the left indicates the storage location; a number on the extreme far left is the entry point of a control transfer. The instruction code, or data, is

[15] The conflicts sparked by these incidents are described beginning on page 122.

[16] Worsley, "Serial Programming for Real and Idealized Digital Calculating Machines", 133.

in the third column between the vertical lines. If enclosed in brackets, this value will be modified by the program as it runs. The fourth column contains comments.

A.3.1 Short Example

This short, nearly trivial, four line program puts |C(004)| in (004), computing the absolute value.

```
        100 | S    004 | -C(004) to Acc.
        102 | C-   106 | test sign
        104 | t    004 | -C(004) to (004) if C(004) < 0
102 →   106 | t    002 | clear Acc.
```

In line 100, the contents of location 004 are subtracted to the accumulator (it is assumed in advance that the accumulator is clear). If the contents of the accumulator are negative – C(004) was a positive number – line 102 causes program control to transfer to line 106 and end the program. If the accumulator is positive – C(004) was negative – then transfer the positive (and absolute) value in the accumulator to location 004, and end the program.

Unsurprisingly for such a straightforward operation, the EDSAC and Mark I programs are similar. All three are four lines long and the EDSAC code is functionally identical to the UTEC.

A.3.2 Extended Arithmetic Example

This longer example computes x^{1024}; x is assumed to be stored in location 004, and the result is also placed in location 004. Notably, this code uses a subroutine to handle the multiplication, which will be explained shortly.

```
         100 | S   202 |  Set counter
126 →    102 | t   002 |
         104 | A   004 |
         106 | T   160 |  Replace C(004)
         110 | t   162 |  by C(004)^2, using
         112 | A   112 |  multiplication          Cycle
         114 | U   156 |  subroutine              ten
         116 | A   164 |                          times
         120 | t   004 |
         122 | A   002 |  Count
         124 | A   200 |  and test
         126 | C-  102 |
             |         |
         200 | A   001 |  Assume
         202 | A   012 |
```

Simply put, this routine computes x^2 10 times, producing x^{1024}. In greater detail, lines 100 and 102 prepare a counter so that the main body, lines 104–120, will cycle 10 (012 in octal) times, a number set in line 202. The cycle is controlled in lines 122–124: the first two of those lines decrements the counter by 1 and line 124 re-enters the cycle unless the counter equals 0. To compute x^2 in each cycle, the main body places x in storage location 160 and 162. These are two special locations, hard coded into the multiplication subroutine to contain the multiplier and multiplicand. The subroutine is stored from location 156 to 306, and is entered using what appears to be a Wheeler jump on lines 112 and 114, though no actual subroutine code exists making verification impossible. The Wheeler jump was well known by the 1950s as one of the most efficient means of enter and return from a subroutine. According to Worsley, the subroutines that existed for UTEC were all of the 'closed' type, a term derived from EDSAC usage. A closed subroutine was input into storage only once, generally at the end of the main program, and called by the main program whenever needed by a special calling sequence, in this case a Wheeler jump.[17] The subroutine used in this case put the product of the multiplication into location 164, which was transferred

[17] A closed subroutine would have been placed within the main body of the program. For more on EDSAC subroutines, see Campbell-Kelly, "Programming the EDSAC: Early Programming Activity at the University of Cambridge", 17–18.

back to location 004 on line 120 so that the next cycle could begin, if necessary.

As both EDSAC and the Mark I had a multiplication operation implemented in hardware, Worsley's versions of the program for those machines was slightly shorter, not having to use a subroutine, although the flow of all three was similar. The EDSAC code is again functionally identical to that of UTEC, while the Mark I code makes appropriate use of one of the B-lines to count the cycles.