NAME

atlas − an 1103A emulator for UNIX systems

SYNOPSIS

atlas

DESCRIPTION

Atlas is an emulator for the Univac Scientific 1103A. In addition to implementing all 41 basic instructions of the CPU, including repeat (RPjnw), Atlas also simulates the program interrupt feature. The emulator runs as an interactive, command line program and interprets binary 1103A machine code stored in simulated core/drum memory. Facilities provided by the emulation include direct entry of machine code in octal form, breakpoints, disassembly, and manual step debugging. Simulated input/output devices include a paper tape reader, a paper tape punch, and an electronic typewriter.

The emulator is written in C and should be portable to a large number of UNIX systems. It requires a compiler that supports two C99 extensions to the C89 standard: 64 bit integers *(unsigned long long)* and C++ -style comments (//). The emulator currently runs on Linux (gcc or Intel cc for IA-32), SunOS (Sun cc or gcc for SPARC), and Darwin (gcc for PowerPC).

Theory of Operation

Atlas implements the machine described in *Univac Scientific General Purpose Computer System Programming* (Remington Rand Univac, September 1956). The 1103A is a 36 bit, word addressed, 1's complement scientific computer featuring a 72 bit accumulator. The fundamental operation of the machine is subtraction of a 36 bit operand from the 72 bit accumulator. The subtractive accumulator avoids the appearance of negative zero for most common 1's complement arithmetic operations. The machine's storage consists of primary magnetic core and secondary magnetic drum storage. The core registers, drum registers, accumulator register, and multiplier-quotient register all appear in a single 15 bit address space.

Machine Instruction Formats

All machine instructions are single word (36 bits). The machine generally uses a two operand instruction format OCuv containing a 6 bit operation code, OC, and two 15 bit addresses, U and V.

A machine instruction is represented in printable form by 12 octal digits OC UUUUU VVVVV. If the 36 bits of the instruction are represented by (*i*_35 ... *i*_0), then OP is (*i*_35 ... *i*_30), U is (*i*_29 ... *i*_15), and V is (*i*_14 ... *i*_0).

Some instructions encode immediate values, and are represented by the format OCjnk, where J is the three bit quantity (*i*_29 ... *i*_27), N is the 12 bit quantity (*i*_26 ... *i*_15), and K is (*i*_14 ... *i*_0).

The emulator represents a single 1103A machine word as the 36 least significant bits of a 64 bit unsigned integer.

Magnetic Core/Drum Storage

The 1103A has three banks of 4096 words of magnetic core storage, and four banks of 4096 words of magnetic drum storage. These storage areas are represented by two globally-accessible arrays of 64 bit integers:

unsigned long long mcs[3*4096];

unsigned long long md[4*4096];

The core and drum memory share a single 15 bit address space. Core addresses range from 00000-27777 octal; drum addresses range from 40000-77777 octal.

The magnetic drum data are persistent; they are stored to disk between invocations of the interpreter. These drum images are binary dumps of the interpreter memory, and are not intended to be portable between implementations.

Arithmetic and Logic Unit

The registers of the ALU are:

AL Accumulator left half register (36 bit, *a*_71 ... *a*_36). Contains the 36 most significant bits of the accumulator.

AR Accumulator right half register (36 bit, *a*_35 ... *a*_0). Contains the 36 least significant bits of the accumulator. AR is a valid operand for most instructions. Any address in the range 32000-32777 octal will specify AR as an operand.

Q Quotient register (36 bit, *q*_35 ... *q*_0). Contains the 36 bit multiplier or quotient during multiply or divide operations. Q is a valid operand for most instructions. Any address in the range 31000-31777 octal will select Q as an operand.

Control Registers

The main control registers for the machine are:

MCR Main control register (6 bit). Holds the operation code of the instruction to be executed.

UAK *u* address counter (15 bit). Holds the address of the *u* operand.

VAK *v* address counter (15 bit). Holds the address of the *v* operand.

PAK Program address counter (15 bit). Contains the address of the next instruction to be executed.

SAR Storage address register (15 bit). Contains the address of the most recently accessed memory register.

X Exchange register (36 bit, *x*_35 ... *x*_0). Provides a data path between the memory, control, and arithmetic unit registers.

F1, F2, F3
Fixed registers (36 bit, *f* _35 ... *f* _0). These 36 bit storage locations, with addresses, 00000, 00001, and 00002 respectively, are used by the control section to store return addresses from jump instructions, subroutine calls, and the program interrupt. F1 is switchable between 00000 in core and 40001 on drum.

Large arithmetic and control registers are represented by the *unsigned long long* data type, while shorter registers are represented by *unsigned int.* The emulator keeps all arithmetic and control registers in global memory.

Machine Cycle

The basic 1103A machine cycle is execute − fetch next. The emulator represents micro-operations as functions without arguments which manipulate the global state of the machine maintained in the simulated core/drum memory registers and arithmetic/control registers. The emulator does not strive for efficiency in implementation; instead it follows the register transfer level description of each operation presented in the original system programming manual.

Atlas catches UNIX signal SIGUSR1 to emulate the program interrupt feature of the 1103A, and jumps to the F3 fixed address (00002 octal) to find the user-provided interrupt handler. As a debugging aid, the user may also stop the machine by sending SIGINT (Ctrl-C) from the console.

The emulated machine cycle is asynchronous; Atlas does not reproduce the actual instruction execution timing of the 1103A hardware.

Input/Output Facilities

Atlas includes simulations of a paper tape reader, a paper tape punch, and an electronic typewriter for character input/output. The 1103A also uses Uniservo magnetic tape and standard 80 column punched cards for word input/output, and a high-speed line printer. Atlas does not provide facilities for word input/output or high-speed printing at this time.

The input/output registers are:

IOA	Input/output register A (8 bit).
IOB	Input/output register B (36 bit). Multi-purposed as both a control register for all input/output devices, and as a word-at-a-time input/output register for word devices such as magnetic tape and punched cards.
HPR	High speed punch register (7 bit). Provides an output-only path to the paper tape punch.
TWR	Electric typewriter register (6 bit). Provides an output-only path to the electric typewriter.

The 1103A uses 7-track paper tape for character input/output. Typically the first six tracks are used for data representation, while the 7th track is used for control (marking word boundaries, etc.). One group of 7 holes is called a tape *frame.* Six consecutive tape frames can be used to hold a 36 bit machine word, with two octal digits stored in the lower 6 tracks of each frame. This is termed *bioctal* tape encoding. The simulated tape reader and punch store tape frames as the low order 7 bits of a standard 8 bit *unsigned char* data type. The tape reels are standard disk files.

In principle, any character set can be used with the machine. In practice, each input/output device usually uses a specific character encoding. The 1103A electronic typewriter is a Commercial Controls Corporation Flexowriter, and uses a unique 6 bit character code. The mapping from this 6 bit code to standard 7 bit ASCII appears in the file *charcode.txt* in the Atlas source distribution. Sending a 6 bit character code to the typewriter register TWR results in its translation to ASCII, and printing of the resulting ASCII character in the interpreter console window, as well as to a disk file. Characters are written to disk using the low order 6 bits of the standard 8 bit *unsigned char* data type.

Emulator Interface

The emulator *main()* routine accepts simple one- or two-word commands using the GNU Readline Library, and changes the machine state accordingly. The **step** command supports single-step debugging. Breakpoints are set using the standard 1103A interface − the manual jump (MJjv) or the manual stop (MSjv) instructions − at appropriate points in the program. The manual stop and jump switches are set using the emulator’s **mj** and **ms** commands.

state	show machine state
start	start machine
step	step machine one instruction
pak <*addr*>	set program counter to octal address <*addr*>
al <*val*>	set accumulator left half to octal value <*val*>

ar <*val*>
set accumulator right half to octal value <*val*>

q <*val*>
set q register to octal value <*val*>

mj <*val*>
toggle manual jump switch, <*val*>= 1|2|3

ms <*val*>
toggle manual stop switch, <*val*>= 1|2|3

<addr> <*val*>
set octal address **<addr>** to octal value <*val*>

show <*addr*>
disassemble memory at octal address <*addr*>

bp <*addr*>
toggle breakpoint at octal address <*addr*>

rew rewind the paper tape reader

f1 toggle F1 switch between 00000 (default) and 40001

help print help information

exit leave program (also Ctrl-D)

Operation Codes

EXAMPLE

In the following example, Atlas uses the repeat RPjnw and print PR-v instructions to type the 6-character message "hello\n" on the electronic typewriter. The message appears both on the emulator console and in the **tw** file.

Program Description

The machine is set to start at address 40002 octal using the **pak** and **start** commands. The machine interprets RPjnw and first stores the low order 15 bits W = 40004 in the low order 15 bits of fixed register F1. Traditionally, a jump instruction MJjv is placed at address F1. The address V of the MJjv instruction at location F1 is therefore modified by RPjnw to specify where to transfer control at the completion of the repeated operation.

The machine next loads the PR-v instruction into the program control registers MCR, UAK, and VAK. The parameter J determines address modification of the repeated instruction. For J=1 in the example, only the V part of the repeated instruction (low order 15 bits) are modified. The PR-v instruction sends the low order 6 bits of address V to the typewriter. The machine performs address modification in the VAK register to step through the six consecutive memory locations 40005 to 40012. In this type of operation, VAK is performing the function of an index register.

At the completion of the repeat instruction, control always transfers to the fixed address F1. Here a traditional jump MJjv transfers control to the address W = 40004 specified as the least significant 15 bits of the RPjnw instruction. The program stop PS-- instruction at address 40004 then causes the machine to halt.

The memory footprint of the example program is ten 36 bit words, 4 to hold operations and 6 to hold data.

Emulator Console Output

```
 $ ./atlas
 Atlas 1103A emulator version 0.1
```

```
loading drum... done
connecting typewriter... done
connecting paper tape punch... done
connecting paper tape reader... done
type "help" for command list
1103A> pak 40002
                  q: 000000000000
 al: 000000000000  ar: 000000000000
                  x: 753000640004
     mcr: 75 uak: 10006 vak: 40004
            pak: 40003 sar: 40002
    f1: 40001  ms: 0 0 0  mj: 0 0 0

addr    value          op
77777   00 00000 00004  ??
40000   00 00000 00000  ??
40001   45 00000 00000  MJjv
40002 ->75 10006 40004  RPjnw
40003   61 00000 40005  PR-v
40004   57 00000 00000  PS--
40005   00 00000 00005  ??
40006   00 00000 00020  ??
40007   00 00000 00011  ??
40010   00 00000 00011  ??
40011   00 00000 00003  ??
40012   00 00000 00045  ??
1103A> start
starting... send SIGINT (Ctrl-C) to stop
hello
PS--: program stop
1103A>
```

INSTALLATION

See the example Makefiles in the source distribution. The emulator uses the GNU Readline Library for command input, so the object file should be linked with *-lreadline.*

FILES

./md	magnetic drum image
./ptr	paper tape reader input
./ptp	paper tape punch output
./tw	typewriter output

TODO

Run CPU diagnostics

Floating point operations

Magnetic tape and punched card I/O

High-speed line printer

1103 address mode

Locate original software systems to run on the emulator
http://hopl.murdoch.edu.au/findlanguages.prx?
id=17&which=byhw&Name=UNIVAC%201103A

Port to Bob Supnik's SIMH simulator framework?

BUGS

Arithmetic errors (overflow, divide by zero, etc.) are not detected

Some jump instructions operating under repeat RPjnw (075) do not behave properly

HISTORY

The 1103A (c. 1956) was one of the first two commercially available, large-scale scientific computers to support hardware floating point arithmetic. The competing machine was the IBM 704, for which FORTRAN and LISP were developed. The 1103A was a revised version of the ERA 1103 (c. 1953). These machines were designed and built by the Engineering Research Associates division of Remington Rand Corporation in St. Paul, Minnesota.

The 1103, or Atlas II, was a follow-on to the ERA 1101, or Atlas, a 24 bit drum memory machine designed and built in the late 1940's under contract with the National Security Agency. The binary representation of the number (Task 13) assigned to the contract became the machine model number.

The code name Atlas was chosen by the NSA in reference to a slide-rule-carrying character from the popular 1940's cartoon *Barnaby* drawn by Crockett Johnson. In the comic strip, Atlas would provide an answer to any question asked of him, but only after first performing a computation on his slide rule.

The history of these machines and the company which designed and built them is documented in:

Erwin Tomash and Arnold A. Cohen, "The Birth of an ERA: Engineering Research Associates, Inc. 1946-1955," *Annals of the History of Computing,* Vol. 1, No. 2, pp. 83-97, Oct. 1979.

Samuel S. Snyder, "Computer Advances Pioneered by Cryptologic Organizations," *Annals of the History of Computing,* Vol. 2, No. 1, pp. 60-70, Oct. 1979.

The 1103A was eventually superseded by the 1107 (c. 1962), from which the Univac 1100 and 2200 series computers are descended.

AUTHOR

Leif Harcke <lharcke@stanford.edu>