An overview of a simplified five-stage pipeline CPU

To
Professor Mikko Lipasti
CS/ECE 552 Introduction to Computer Architecture

By
Peng Cheng, Yiming Liu, Yi Shen
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
May 2nd, 2018

Overview

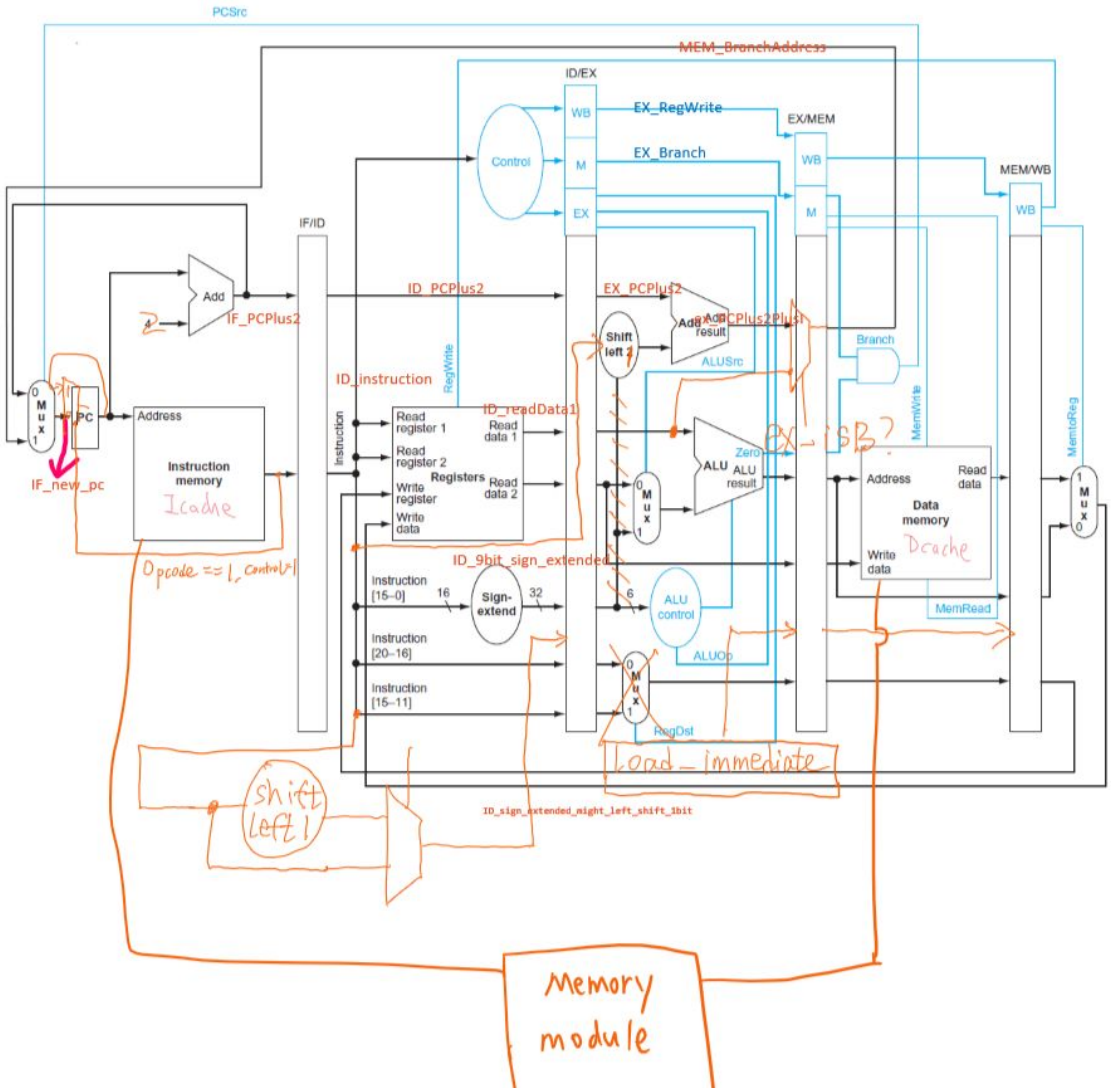
For this entire project, we accomplished the design and testing of a pipelined, cached, 5-stage microprocessor. This process has a bitwidth of 16 bits, cache and memory byte addressable. The processor uses instruction set WISC-S18.

In our design phase 1, we designed a basic single cycle MIPS like processor, it contained no support for data forwarding, stall, and memory module is single cycle.

For our design phase 2, we added pipeline into our phase 1 design, supported data forwarding and global stall signal. The process was able to run at a higher frequency, even though we didn't test it.

For our last phase, cache system was added. We no longer have 2 seperate memory module, one for instruction and one for data. They are each replaced with a cache module, and both connecting to the 4-cycle memory module.

The following is our final design diagram.



Responsibility / Task Breakdown:

We incorporated pair programming throughout our 3 design and test phases, this is to ensure the maximize communication efficiency and reduce human error during our design phase.

Following is the table detailing each team member's contribution for each phases:

Phase	Yi Shen	Yiming Liu	Peng Cheng
Phase 1	33%	33%	33%
Phase 2	37.5%	37.5%	25%
Phase 3	45%	45%	10%

Special Features

In phase 2, a single cycle processor is upgraded to a five-stage processor, which also supports data forwarding and register bypassing. A typical scenario in a pipelined processor that demands data forwarding are known as register read after write (RAW) and register write after write (WAW). To figure out whether a register is modified and then being read or written before being written back, which is accomplished in the fifth stage in our design, we first need to check if an instruction involves register modification. In other words, if an instruction does not modify any register, it is unnecessary to do a hazard detection on it. Hence, based on the provided instruction set architecture (ISA) by professor Mikko Lipasti, we decode each instruction at IF/ID stage to save registers that will be modified in later stages to IF/ID pipeline stages. Three signals IF_RsExists, IF_RtExists and IF_RdExists are defined correspondingly in IF stage and also passed to later stages as input signal to hazard detection logic. Furthermore, we implement stall detection unit between IF stage and ID stage so that only IF stage will be stalled if data forwarding cannot be achieved. Besides, since branch unit is designed under assumption that branch will always always be taken, we flush all instructions that enter the processor after the branch instruction and all related control signals by writing zero to specific pipeline registers.

In phase 3, instructions and data is read from cache instead of memory when there is a hit. However, a single 2 bytes data or instruction fetching from memory costs four clock cycles and 16 bytes in total is put in a cache block after each misses. Considering eight two-byte data is read from memory, we decide to do a pipelined reading from memory to reduce cache miss penalty. Instead of sending a new memory address to memory after a valid data is returned, which means sending a address every four clock cycle, we send an address in each cycle for eight times. In this case, memory is able to manage sending valid data back to cache and taking

addresses from cache concurrently. Consequently, delay on memory access reduces from 32 cycles to 12 cycles. Moreover, there might be a conflict between instruction cache and data cache if they compete for memory access simultaneously. An arbitration module is created to allow only one cache takes over the memory in a clock cycle and the other one is stalled accordingly. Data cache is rendered a higher priority in our design. If one of those two caches is denoted as busy fetching data, the other one cannot read from memory until the fetching is completed. Lastly, the whole processor is stalled if a cache miss occurs.

Completeness

Our design in phase1, phase2 and phase3 meets all the specified requirements. For phase3, we successfully passed all the provided four tests. During the demo, we successfully showed professor Lipasti and TA that our design functions as expected. Simulation results is shown and explained in Results section.

Testing

1. Analyzing the waveform and tracing the related signals are our main testing and debugging strategies. If we noticed that a certain signal was not generated correctly, we would check a set of signals that determines the erroneous one and analyze those more basic pieces first.
2. We used unit testing in phase 3 for the instruction cache and data cache; specifically, since we were given the memory unit in phase 2, we replaced part of the cache with the given one and see if other parts worked as expected and continued to do so until find the error.
3. We ran into a problem where a bunch of signals turned out to be high-Z. To solve it, we targeted the first signal that went to high-Z. Also, we sometimes gave those signals constants to see if other signals that depend upon it worked as expected.
4. For the fourth test in phase 3, each team member worked out the result of each instruction by hand and compared the results in registers with the contents from our design to help us find the bug.

Results

- Phase1

Test Number	sim_cycles	inst_count	Result

1	14	11	Correct
2	15	12	Correct
3	22	19	Correct

Registers Values:

Test1:
PC: 0x0000 REG: 1 VALUE: 0x0051
PC: 0x0002 REG: 1 VALUE: 0x5151
PC: 0x0004 REG: 2 VALUE: 0x00b0
PC: 0x0006 REG: 2 VALUE: 0xa0b0
PC: 0x0008 REG: 3 VALUE: 0xf201
PC: 0x000a REG: 4 VALUE: 0xb0a1
PC: 0x000c REG: 6 VALUE: 0x42a0
PC: 0x000e REG: 7 VALUE: 0x0000
PC: 0x0010 REG: 8 VALUE: 0x10a8
PC: 0x0012 REG: 9 VALUE: 0x2a04
PC: 0x0014

Test2:
PC: 0x0000 REG: 1 VALUE: 0x0051
PC: 0x0002 REG: 1 VALUE: 0x5151
PC: 0x0004 REG: 2 VALUE: 0x00b0
PC: 0x0006 REG: 2 VALUE: 0x00b0
PC: 0x0008 REG: 3 VALUE: 0x0004
PC: 0x000a REG: 3 VALUE: 0x0004
PC: 0x000c ADDR: 0x00b4 VALUE: 0x5151
PC: 0x000e REG: 5 VALUE: 0x00b4
PC: 0x0010 REG: 4 VALUE: 0x5151 ADDR: 0x00b4
PC: 0x0012 REG: 5 VALUE: 0x5101
PC: 0x0014 REG: 5 VALUE: 0x000e
PC: 0x0016

Test3:
PC: 0x0000 REG: 1 VALUE: 0x0002
PC: 0x0002 REG: 1 VALUE: 0x0002
PC: 0x0004 REG: 2 VALUE: 0x0001
PC: 0x0006 REG: 2 VALUE: 0x0001
PC: 0x0008 REG: 6 VALUE: 0x0004

PC: 0x000a REG: 6 VALUE: 0x0004
PC: 0x000c REG: 1 VALUE: 0x0001
PC: 0x000e REG: 5 VALUE: 0x0010
PC: 0x0010
PC: 0x0012
PC: 0x0004 REG: 2 VALUE: 0x0001
PC: 0x0006 REG: 2 VALUE: 0x0001
PC: 0x0008 REG: 6 VALUE: 0x0004
PC: 0x000a REG: 6 VALUE: 0x0004
PC: 0x000c REG: 1 VALUE: 0x0000
PC: 0x000e REG: 5 VALUE: 0x0010
PC: 0x0010
PC: 0x0016 REG: 4 VALUE: 0x0005
PC: 0x0018

- Phase2

Test Number	sim_cycles	inst_count	Results
1	18	11	Correct
2	20	11	Correct
3	32	16	Correct

Registers Values:

Test1:
REG: 1 VALUE: 0x0051
REG: 1 VALUE: 0x5151
REG: 2 VALUE: 0x00b0
REG: 2 VALUE: 0xa0b0
REG: 3 VALUE: 0xf201
REG: 4 VALUE: 0x7fff
REG: 6 VALUE: 0x8dfe
REG: 7 VALUE: 0x0000
REG: 8 VALUE: 0xe37f
REG: 9 VALUE: 0xdff8

Test2:

REG: 1 VALUE: 0x0051
REG: 1 VALUE: 0x5151
REG: 2 VALUE: 0x00b0
REG: 2 VALUE: 0x00b0
REG: 3 VALUE: 0x0004
REG: 3 VALUE: 0x0004
STORE: ADDR: 0x00b4 VALUE: 0x5151
REG: 5 VALUE: 0x00b4
LOAD: ADDR: 0x00b4 VALUE: 0x5151
REG: 4 VALUE: 0x5151
REG: 5 VALUE: 0x5101
REG: 5 VALUE: 0x000e

Test3:
REG: 1 VALUE: 0x0002
REG: 1 VALUE: 0x0002
REG: 2 VALUE: 0x0001
REG: 2 VALUE: 0x0001
REG: 6 VALUE: 0x0004
REG: 6 VALUE: 0x0004
REG: 1 VALUE: 0x0001
REG: 5 VALUE: 0x0010
REG: 2 VALUE: 0x0001
REG: 2 VALUE: 0x0001
REG: 6 VALUE: 0x0004
REG: 6 VALUE: 0x0004
REG: 1 VALUE: 0x0000
REG: 5 VALUE: 0x0010
REG: 4 VALUE: 0x0005

- Phase3

Test Number	sim_cycles	inst_count	Result
1	42	11	Correct
2	57	12	Correct
3	56	16	Correct
4	359	119	Correct

Registers Values:

Test1:

LOAD: ADDR: 0x0000 VALUE: 0x0000
LOAD: ADDR: 0x0002 VALUE: 0x0000
LOAD: ADDR: 0x0004 VALUE: 0x0000
LOAD: ADDR: 0x0006 VALUE: 0x0000
LOAD: ADDR: 0x0008 VALUE: 0xb151
LOAD: ADDR: 0x000a VALUE: 0xa151
LOAD: ADDR: 0x000c VALUE: 0xb2b0
LOAD: ADDR: 0x000e VALUE: 0xa2a0
REG: 1 VALUE: 0x0051
REG: 1 VALUE: 0x5151
REG: 2 VALUE: 0x00b0
REG: 2 VALUE: 0xa0b0
LOAD: ADDR: 0x0010 VALUE: 0x0000
LOAD: ADDR: 0x0012 VALUE: 0x0000
LOAD: ADDR: 0x0014 VALUE: 0x0000
LOAD: ADDR: 0x0016 VALUE: 0x0000
LOAD: ADDR: 0x0018 VALUE: 0x5862
LOAD: ADDR: 0x001a VALUE: 0x698a
LOAD: ADDR: 0x001c VALUE: 0xf000
LOAD: ADDR: 0x001e VALUE: 0xxxxx
REG: 3 VALUE: 0xf201
REG: 4 VALUE: 0x7fff
REG: 6 VALUE: 0x8dfe
REG: 7 VALUE: 0x0000
REG: 8 VALUE: 0xe37f
REG: 9 VALUE: 0xdff8

Test2:

LOAD: ADDR: 0x0000 VALUE: 0x0000
LOAD: ADDR: 0x0002 VALUE: 0x0000
LOAD: ADDR: 0x0004 VALUE: 0x0000
LOAD: ADDR: 0x0006 VALUE: 0x0000
LOAD: ADDR: 0x0008 VALUE: 0xb151
LOAD: ADDR: 0x000a VALUE: 0xa151
LOAD: ADDR: 0x000c VALUE: 0xb2b0
LOAD: ADDR: 0x000e VALUE: 0xa200
REG: 1 VALUE: 0x0051
REG: 1 VALUE: 0x5151
REG: 2 VALUE: 0x00b0
REG: 2 VALUE: 0x00b0

LOAD: ADDR: 0x0010 VALUE: 0x0000
LOAD: ADDR: 0x0012 VALUE: 0x0000
LOAD: ADDR: 0x0014 VALUE: 0x0000
LOAD: ADDR: 0x0016 VALUE: 0x0000
LOAD: ADDR: 0x0018 VALUE: 0x8450
LOAD: ADDR: 0x001a VALUE: 0x7542
LOAD: ADDR: 0x001c VALUE: 0x2555
LOAD: ADDR: 0x001e VALUE: 0xf000
REG: 3 VALUE: 0x0004
LOAD: ADDR: 0x00b0 VALUE: 0x0000
LOAD: ADDR: 0x00b2 VALUE: 0x0000
LOAD: ADDR: 0x00b4 VALUE: 0x0000
LOAD: ADDR: 0x00b6 VALUE: 0x0000
LOAD: ADDR: 0x00b8 VALUE: 0xxxxx
LOAD: ADDR: 0x00ba VALUE: 0xxxxx
LOAD: ADDR: 0x00bc VALUE: 0xxxxx
LOAD: ADDR: 0x00be VALUE: 0xxxxx
LOAD: ADDR: 0x00b4 VALUE: 0x0000
STORE: ADDR: 0x00b4 VALUE: 0x5151
REG: 3 VALUE: 0x0004
REG: 5 VALUE: 0x00b4
REG: 4 VALUE: 0x5151
REG: 5 VALUE: 0x5101
REG: 5 VALUE: 0x000e

Test3:
LOAD: ADDR: 0x0000 VALUE: 0x0000
LOAD: ADDR: 0x0002 VALUE: 0x0000
LOAD: ADDR: 0x0004 VALUE: 0x0000
LOAD: ADDR: 0x0006 VALUE: 0x0000
LOAD: ADDR: 0x0008 VALUE: 0xb102
LOAD: ADDR: 0x000a VALUE: 0xa100
LOAD: ADDR: 0x000c VALUE: 0xb201
LOAD: ADDR: 0x000e VALUE: 0xa200
REG: 1 VALUE: 0x0002
REG: 1 VALUE: 0x0002
REG: 2 VALUE: 0x0001
REG: 2 VALUE: 0x0001
LOAD: ADDR: 0x0010 VALUE: 0x0000
LOAD: ADDR: 0x0012 VALUE: 0x0000
LOAD: ADDR: 0x0014 VALUE: 0x0000

LOAD: ADDR: 0x0016 VALUE: 0x0000
LOAD: ADDR: 0x0018 VALUE: 0xc202
LOAD: ADDR: 0x001a VALUE: 0xde60
LOAD: ADDR: 0x001c VALUE: 0xf000
LOAD: ADDR: 0x001e VALUE: 0x0462
REG: 6 VALUE: 0x0004
REG: 6 VALUE: 0x0004
REG: 1 VALUE: 0x0001
REG: 5 VALUE: 0x0010
REG: 2 VALUE: 0x0001
REG: 2 VALUE: 0x0001
REG: 6 VALUE: 0x0004
REG: 6 VALUE: 0x0004
REG: 1 VALUE: 0x0000
REG: 5 VALUE: 0x0010
REG: 4 VALUE: 0x0005

Test4:
LOAD: ADDR: 0x0000 VALUE: 0x0000
LOAD: ADDR: 0x0002 VALUE: 0x0000
LOAD: ADDR: 0x0004 VALUE: 0x0000
LOAD: ADDR: 0x0006 VALUE: 0x0000
LOAD: ADDR: 0x0008 VALUE: 0xb18a
LOAD: ADDR: 0x000a VALUE: 0xb680
LOAD: ADDR: 0x000c VALUE: 0xa601
LOAD: ADDR: 0x000e VALUE: 0x9160
REG: 1 VALUE: 0x008a
REG: 6 VALUE: 0x0080
LOAD: ADDR: 0x0180 VALUE: 0x0000
LOAD: ADDR: 0x0182 VALUE: 0x0000
LOAD: ADDR: 0x0184 VALUE: 0x0000
LOAD: ADDR: 0x0186 VALUE: 0x0000
LOAD: ADDR: 0x0188 VALUE: 0xxxxx
LOAD: ADDR: 0x018a VALUE: 0xxxxx
LOAD: ADDR: 0x018c VALUE: 0xxxxx
LOAD: ADDR: 0x018e VALUE: 0xxxxx
LOAD: ADDR: 0x0180 VALUE: 0x0000
STORE: ADDR: 0x0180 VALUE: 0x008a
REG: 6 VALUE: 0x0180
LOAD: ADDR: 0x0010 VALUE: 0x0000
LOAD: ADDR: 0x0012 VALUE: 0x0000
LOAD: ADDR: 0x0014 VALUE: 0x0000

LOAD: ADDR: 0x0016 VALUE: 0x0000
LOAD: ADDR: 0x0018 VALUE: 0x0223
LOAD: ADDR: 0x001a VALUE: 0x1113
LOAD: ADDR: 0x001c VALUE: 0xc1fc
LOAD: ADDR: 0x001e VALUE: 0xb110
REG: 1 VALUE: 0x0010
REG: 2 VALUE: 0x0070
LOAD: ADDR: 0x0070 VALUE: 0x0000
LOAD: ADDR: 0x0072 VALUE: 0x0000
LOAD: ADDR: 0x0074 VALUE: 0x0000
LOAD: ADDR: 0x0076 VALUE: 0x0000
LOAD: ADDR: 0x0078 VALUE: 0xxxxx
LOAD: ADDR: 0x007a VALUE: 0xxxxx
LOAD: ADDR: 0x007c VALUE: 0xxxxx
LOAD: ADDR: 0x007e VALUE: 0xxxxx
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x0010
REG: 3 VALUE: 0x0002
REG: 2 VALUE: 0x0072
REG: 1 VALUE: 0x000e
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x000e
REG: 2 VALUE: 0x0074
REG: 1 VALUE: 0x000c
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x000c
REG: 2 VALUE: 0x0076
REG: 1 VALUE: 0x000a
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x000a
REG: 2 VALUE: 0x0078
REG: 1 VALUE: 0x0008
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x0008
REG: 2 VALUE: 0x007a
REG: 1 VALUE: 0x0006
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x0006
REG: 2 VALUE: 0x007c
REG: 1 VALUE: 0x0004
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x0004
REG: 2 VALUE: 0x007e

REG: 1 VALUE: 0x0002
LOAD: ADDR: 0x0070 VALUE: 0x0000
STORE: ADDR: 0x0070 VALUE: 0x0002
REG: 2 VALUE: 0x0080
REG: 1 VALUE: 0x0000
REG: 1 VALUE: 0x0010
LOAD: ADDR: 0x0020 VALUE: 0x0000
LOAD: ADDR: 0x0022 VALUE: 0x0000
LOAD: ADDR: 0x0024 VALUE: 0x0000
LOAD: ADDR: 0x0026 VALUE: 0x0000
LOAD: ADDR: 0x0028 VALUE: 0x9450
LOAD: ADDR: 0x002a VALUE: 0x0553
LOAD: ADDR: 0x002c VALUE: 0x1113
LOAD: ADDR: 0x002e VALUE: 0xc1fa
REG: 2 VALUE: 0x0070
REG: 5 VALUE: 0x0080
REG: 4 VALUE: 0x0010
LOAD: ADDR: 0x0080 VALUE: 0x0000
LOAD: ADDR: 0x0082 VALUE: 0x0000
LOAD: ADDR: 0x0084 VALUE: 0x0000
LOAD: ADDR: 0x0086 VALUE: 0x0000
LOAD: ADDR: 0x0088 VALUE: 0xxxxx
LOAD: ADDR: 0x008a VALUE: 0xxxxx
LOAD: ADDR: 0x008c VALUE: 0xxxxx
LOAD: ADDR: 0x008e VALUE: 0xxxxx
LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x0010
REG: 2 VALUE: 0x0072
REG: 5 VALUE: 0x0082
REG: 1 VALUE: 0x000e
REG: 4 VALUE: 0x000e
LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x000e
REG: 2 VALUE: 0x0074
REG: 5 VALUE: 0x0084
REG: 1 VALUE: 0x000c
REG: 4 VALUE: 0x000c
LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x000c
REG: 2 VALUE: 0x0076
REG: 5 VALUE: 0x0086
REG: 1 VALUE: 0x000a
REG: 4 VALUE: 0x000a

LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x000a
REG: 2 VALUE: 0x0078
REG: 5 VALUE: 0x0088
REG: 1 VALUE: 0x0008
REG: 4 VALUE: 0x0008
LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x0008
REG: 2 VALUE: 0x007a
REG: 5 VALUE: 0x008a
REG: 1 VALUE: 0x0006
REG: 4 VALUE: 0x0006
LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x0006
REG: 2 VALUE: 0x007c
REG: 5 VALUE: 0x008c
REG: 1 VALUE: 0x0004
REG: 4 VALUE: 0x0004
LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x0004
REG: 2 VALUE: 0x007e
REG: 5 VALUE: 0x008e
REG: 1 VALUE: 0x0002
REG: 4 VALUE: 0x0002
LOAD: ADDR: 0x0080 VALUE: 0x0000
STORE: ADDR: 0x0080 VALUE: 0x0002
REG: 2 VALUE: 0x0080
REG: 5 VALUE: 0x0090
REG: 1 VALUE: 0x0000
LOAD: ADDR: 0x0030 VALUE: 0x0000
LOAD: ADDR: 0x0032 VALUE: 0x0000
LOAD: ADDR: 0x0034 VALUE: 0x0000
LOAD: ADDR: 0x0036 VALUE: 0x0000
LOAD: ADDR: 0x0038 VALUE: 0xc00d
LOAD: ADDR: 0x003a VALUE: 0x0553
LOAD: ADDR: 0x003c VALUE: 0x1113
LOAD: ADDR: 0x003e VALUE: 0xc1fa
REG: 1 VALUE: 0x0010
REG: 5 VALUE: 0x0080
REG: 4 VALUE: 0x0010
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x0082
REG: 1 VALUE: 0x000e

REG: 4 VALUE: 0x000e
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x0084
REG: 1 VALUE: 0x000c
REG: 4 VALUE: 0x000c
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x0086
REG: 1 VALUE: 0x000a
REG: 4 VALUE: 0x000a
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x0088
REG: 1 VALUE: 0x0008
REG: 4 VALUE: 0x0008
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x008a
REG: 1 VALUE: 0x0006
REG: 4 VALUE: 0x0006
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x008c
REG: 1 VALUE: 0x0004
REG: 4 VALUE: 0x0004
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x008e
REG: 1 VALUE: 0x0002
REG: 4 VALUE: 0x0002
REG: 0 VALUE: 0x0000
REG: 5 VALUE: 0x0090
REG: 1 VALUE: 0x0000
LOAD: ADDR: 0x0040 VALUE: 0x0000
LOAD: ADDR: 0x0042 VALUE: 0x0000
LOAD: ADDR: 0x0044 VALUE: 0x0000
LOAD: ADDR: 0x0046 VALUE: 0x0000
LOAD: ADDR: 0x0048 VALUE: 0xb680
LOAD: ADDR: 0x004a VALUE: 0xa600
LOAD: ADDR: 0x004c VALUE: 0x8260
LOAD: ADDR: 0x004e VALUE: 0xb110
REG: 2 VALUE: 0x008a
REG: 1 VALUE: 0x008a
REG: 0 VALUE: 0x0000
REG: 6 VALUE: 0x0180
REG: 6 VALUE: 0x0080
REG: 2 VALUE: 0x0010
REG: 1 VALUE: 0x0010

REG: 0 VALUE: 0x0000
LOAD: ADDR: 0x0050 VALUE: 0x0000
LOAD: ADDR: 0x0052 VALUE: 0x0000
LOAD: ADDR: 0x0054 VALUE: 0x0000
LOAD: ADDR: 0x0056 VALUE: 0x0000
LOAD: ADDR: 0x0058 VALUE: 0x0bba
LOAD: ADDR: 0x005a VALUE: 0x0bba
LOAD: ADDR: 0x005c VALUE: 0x0bba
LOAD: ADDR: 0x005e VALUE: 0xb1aa
REG: 1 VALUE: 0x00aa
REG: 1 VALUE: 0xaaaa