everything method, each state stores: PC, operation, address1, address2, address3, register and memory. Thus, the amount of information to be stored of every states in the bubble sort program and the matrix multiplication program are in total of 63,756 and 13,090 words respectively. It requires a large amount of storage to store the information in every states. However, storing all the states of execution enables users to query any state later without spending time to re-execution. For the reverse computing method, it is not necessary to store all of information in every states because each state stores only information in some particular state at some particular time. Moreover, the reverse arithmetic operation in the reverse computing method can reduce the amount of information that must be stored. Therefore, the information that needs to be stored in the reverse computing method is less than those of the save everything method.

TABLE II. MEASUREMENT OF TWO RUNS: NO-SAVE AND REVERSE COMPUTING (UNIT: WORD, TIME: CLOCK)

Program	No-Save		Reverse Computing	
	Amount of data	*Time to rerun*	*Amount of data*	*Time to rerun*
Bubble sort	0	2,024	51,643	6
Matrix multiplication	0	2,026	10,297	4

In Table II, the no-save method will not store the information of any state but it has to execute the program from the beginning to the state that users want to observe the output. Thus, the amount of information that needs to be stored of both programs is zero. However, it requires time to re-execute the program. The bubble sort program consumes time to re-execute as much as 2,024 clocks and the matrix multiplication program consumes time to re-execute 2,026 clocks. The time to re-execute is dependent on the number of clock that users want to observe. When executing to a very distant clock, seeing the output of the previous clock will definitely take long time to re-execute. Moreover, re-execution wastes the time because it is a executing of a previous state that has already been executed. The reverse computing method will run the reverse operation of the operation in the current state to go-back to any previous time step without starting from the beginning. Therefore, the amount of time to re-execute is equal to the amount of time to execute the operation in the current state, as shown in Fig.1 and Fig.7.

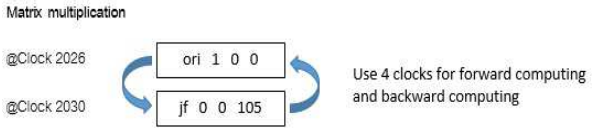

Fig. 7. The backward execution from clock at 2030 to the previous clock in matrix multiplication program

From both programs, the proposed method can reduce the amount of information that will be stored by 20.17% compared to the save-everything method. It also reduces the time by 99.75% compared to the run-from-beginning method.

VII. CONCLUSION

We have developed a reverse operation of data and arithmetic operation and demonstrated through a working implementation that it is possible to build a debugging tool by reverse computing. The method of reverse computing allows a debugger to locate and move back to any states without starting from the beginning and without storing a large amount of state data.

The efficiency of our scheme is illustrated by comparing it with two naïve implementations. First implementation is the method that save all states of all instructions executed. Second implementation is the method that starts its execution from time-zero to the n-1 th instruction. In the first scheme, a large amount of states will be stored. Any state from the beginning to the breakpoint can be accessed and displayed. In the second scheme, no states is stored but it requires to run the program from time-zero to n-1 th instruction. Two simple programs are used as benchmarks: bubble sort and matrix multiplication. All of these three modes were tested in the situation that users chose to track down the output and then step back to the previous clock. The proposed method can reduce the amount of information that will be stored by 20.17% compared to the save-everything method. It also reduces the time by 99.75% compared to the run-from-beginning method.

The proposed debugging tool can be used as a traditional forward movement debugging. It also can be performed with equal ease in the reverse direction. Moreover, the tool is based on web interface so it would be most convenient way to perform debugging on any platform that supports web interface.

REFERENCES

[1] Axelsen H.B., Gluck R. "What do reversible programs compute?" (2011) Lecture Notes in Computer Science, 6604 LNCS, pp. 42-56.

[2] Stoddart B., Lynas R., Zeyda F., "A Virtual Machine for Supporting Reversible Probabilistic Guarded Command Languages," (2009) Electronic Notes in Theoretical Computer Science, 253 (6), pp. 33-56.

[3] Yokoyama T., "Reversible Computation and Reversible Programming Languages," (2009) Electronic Notes in Theoretical Computer Science, 253 (6), pp. 71-81.

[4] A. More, and J. R. Kumar, V.G., "Web Based Programming Assistance Tool for Novices," (2011) IEEE International Conference on Technology for Education, Chennai, Tamil Nadu.

[5] Lewis B., Ducasse M. Using events to debug Java programs backwards in time (2003) Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA, pp. 96-97.

[6] Boothe B. Efficient algorithms for bidirectional debugging (2000) SIGPLAN Notices (ACM Special Interest Group on Programming Languages), 35 (5), pp. 299-310.

[7] Engblom J. A review of reverse debugging (2012) Proceedings of the Conference on System, Software, SoC and Silicon Debug, art. no. 6338149, .