ECE 697CE project 3 description write-up

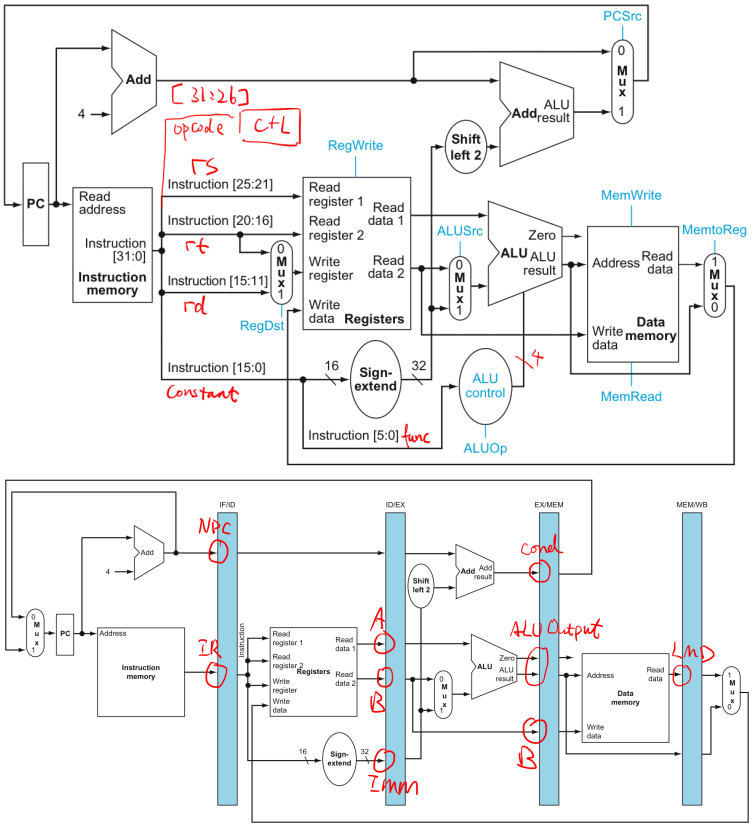
Hao Shi

Prelude

The purpose of this project is to simulate the behavior of a 5-stage MIPS processor by reading in the machine code and display the content of latches, registers and memory as requested by users. Implementation have been done in C++ language, branching and pipelining have been done and no data-forwarding and out-of-order required.

Way of approaching

To design a simulator to mimic a hardware, I have been studied the hardware itself and try to replicate each module in the hardware design into software. Following two figures influenced when designing the simulator.



I first implemented the ALU, then ALU control, register file, memory, PC, finally stage transaction latches as suggested in the hint. The core and most complex part of the simulator is the controller which coordinates all software implemented hardware modules. This simulator puts no faith on implement all the required hardware modules in real processor, which means some part of those modules are merely implemented in compiler features. For example, sign extend is just a type conversion as follows.

```
return (int32_t) num; // the compiler will do it for us
```

Multiplexer are if or switch statements in C++ and wire range are shifting and logical & operations.

```
return (field) (this->inst >> 21u & 0b11111u); // rs
```

In hardware, data are transmitted in wired. To reflect this behaviour, I use the most matching data types in C++ for each input and outputs. For example, register files is an array of 32-bit unsigned integers and ALU output and its latch is 64-bit unsigned integer, instruction field for register is 8-bit and immediate is 16-bit. This restriction helped me on identifying design flaws which can be detected by C++ compiler in very early stage.

For debugging convenience purpose, I have also implemented assembler and disassembler. Those two parts only supports limited instructions as seen in the ASM example file. When running the simulation, the simulator will display the disassembler result to help identify false results. The implementation of the assembler is merely 200 lines of C++ code, in deterministic finite automaton fashion. Disassembler is much easier.

Operation guide

To run the simulation, the program should been started with the ASM file name parameter as its input. Please note that because the program includes assembler, content of the input file is ASM code. In the beginning of the programming that have not been run any simulation, the assembler will translate the ASM code and display its result in hexadecimal, binary and disassembled results, for example:

```
IN  < LUI $t2 0x0000A
HEX > 0x3c0a000a
BIN > 00111100000010100000000000001010
ASM > LUI $t2 0xa
```

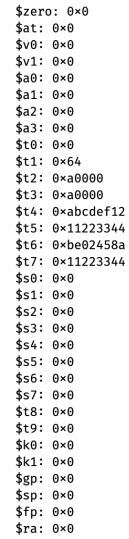
Note that the actual data feed into the simulator is equivalent to that required in the assignment, which is binary presentation of the instruction.

After the assembler the simulator will ask if user want to run those instructions in instruction mode or cycle mode. For debugging and presentation purpose, my simulation also provide burst mode, which can run the program without any interactions and display the results in the end. All of those 3 modes are running in pipeline, which means even in instruction mode, users can still see what instructions are in the pipeline latches. After mode selection, users will be asked to input number of instructions in the file they want to run, or simply input 0 to run all of them. Then, the simulation will begin.

After each step of the simulation (depends on modes except burst), the simulator will display PC, control stall status, number of clock cycle, pipeline utilization efficiency so far and all latches contents.

```
Have run: LUI $t1 0×1234
=====================================
Program Counter: 3
Control Stall: 0
Clock Cycle: 5
Efficiency: 32%
------------------IF-ID---------------
Ins: LUI $t2 0×0
NPC: 2
------------------ID-EX---------------
Ins: ORI $t1 $t1 0×5678
NPC: 1
A: 0×12340000
B: 0×0
I: 0×5678
------------------EX-MEM--------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
=====================================
'R' for registers and continue, anything else to continue…
```

At this point, user can either decide to display register contents or continue. Input ‘R’ the simulator will print all value of all registers, as shown below.
In any case, the simulation will continue. Users are free to decide whether the register should been displayed after any step. In burst mode, the simulator will display registers and data memory right after the simulator ends.



After one round of simulation ended, the simulator will ask users of they want to run the simulation again, if yes the simulator will lead users to mode selection, no the simulator will exit.

Implementation Details

There are some small problems encountered when implementing this simulator. Here is some of them and their solutions chosen in this simulator.

Mimic pipeline

In real processors, pipeline stages are running in parallel. But it is not possible to achieve this safely in software simulator. Instead, this simulator run each stage backwards started from the last stage. This is because after each stage have been run, it result for next stage will be written to the latch between this stage and next stage. If we run the pipeline stages forward from left, those latch will be overwritten before they can be used.

Customized MUL instruction

As the additional instruction required in this assignment, the multiply instruction caused some problem and reflected the reason why the original MUL have such strange pattern. The original MIPS instruction put the result of MUL in 2 special registers each of them are 32-bit and provided a special instruction to copy them into the program registers. In the customized version, results are written in one clock cycle. This will require the ALU to provide 64-bit output specially for MUL. In real world additional 32-bits of wiring and stage latches are wasted if user don’t use this instruction at all. There are also additional shifting operations in WB stage to extract the high most part. Note that in hardware design, WB also needs to write 2 values into the register file in one clock cycle, which can also be difficult. This instruction also require to take care of the data hazard for the additional register, which can also become wasteful for this specific uncommonly used instruction. So, this kind of instruction is not practical and should be avoided in real world.

Branching

If we don’t need branch prediction, branching will be straight forward. When the ID stage detected BEQ instruction, it will put a global stall flag in the pipeline that will be checked by IF and ID stage. When IF stage found this flag have been set, it will do nothing in this cycle, but every other stages will not been affected. In practice if ID stage found this flag have been set, it will ignore its current latch (IF/ID) input and set it NOP; because the simulator runs those stages backwards sequentially this procedure is not required. Value of new PC will be set at MEM stage if the ALU outputs zero, which indicates equal (ALU operation is subtract), the stall flag will also been unset in this stage. This allows the IF stage to take the new PC and continue without waiting for the BEQ instruction to finish its WB stage.

The LUI instruction

LUI is one of the special case that should be handled carefully for it takes an immediate as first input of ALU and shift with specific number of bits (16 bits) as the second input of ALU. This case was taken care of in EXE stage regardless rt field. Everything else is identical to other immediate typed instructions.

SLL and SRL instruction

These 2 instruction are the only instructions that use shamt field and take rt as the first input of the ALU. As suggested implicitly in the figure this was handled in ID stage.

Instructions without write back

BEQ and SW are the only 2 of the instructions that does not have register write back functionality. For those 2 instructions WB stage should ignore them.

LW instruction

This instructions load memory data into register. In WB stage, instead of taking the output from ALU, it takes LMD instead. This is also handled in WB stage.

Data hazard

When control hazard was handled by setting stall flags in the pipeline for IF and ID stages, data hazard is also handled in ID stage. Data hazard can be detected by looking for any instruction register fields feed into EXE stage, which their destination field in following stage latches. When such field is detected, the ID stage will do nothing, preserving the IF/ID latch data and inform the controller that it should not process the following IF stage. In this case IF and ID still keep unchanged in following clock cycle until the stalling instruction have been finished its final stage.

Structural hazard

Even if we are building a software simulator, which we can change multiple fields in one memory component safely, we still need to keep in mind that in hardware this will cause structural hazard. In this simulator, structural hazard have been avoided by using 2 memory unit for data and instructions. Even if IF and MEM stage collide, it will be fine for them and stalling are not required.

The assembler

Comparing with some parser and lexer I had worked with, assembler is the one of the most easy kind of them. The assembler in this simulator is a NFA. If first read in string tokens that contains only A-Z characters. Then the assembler will try to match the string with any rules hardcoded to decide what kind of instruction parameters it should read. Currently, there are only 2 kinds of instruction parameters supported, registers and hexadecimal numers for immediate. For example, if the assembler detected the line it read have leading BEQ, it will try to read 2 register and then a hexadecimal number, then it will use bit shifting and logical or operation to assemble the instruction.

```
uint32_t s = read_reg(code, cursor);
uint32_t t = read_reg(code, cursor);
uint32_t i = read_imme(code, cursor);
ins = ins | s << 21u | t << 16u | i;
```

The disassembler

When we know how to decode instructions for ID stage, disassembler is more straight forward. It first detect the instruction name and then extract parameters to concatenate into ASM string.

Code style

C++ is one of the most long-lived programming languages that have lots of pitfalls and unsafe usage, for example memory leaks and dangling pointers. It also provides some useful tools to avoid those pitfalls. The program guideline of this project is do make use of modern C++ practice and avoid using pointers. With these principles kept in mind combine with some tools, I have not encountered any difficulties in the process.

Conclusion

In this simulator project, I have reached my primary objective and also built some tools to prove its correctness. The result output of the ASM code have been verified and it works just fine. All features, including branching, pipelining, various of hazard avoidance and all instructions.

Appendix: Runtime output example

```
/Users/shisoft/Documents/IdeaProjects/ECE697/MIPSim/cmake-build-debug/Project_3
MIPSInstruction.s
UMass Amherst ECE 697CE Project 3 - Hao Shi
Reading MIPS code file MIPSInstruction.s
IN  < LUI $t1 0x1234

HEX > 0x3c091234
BIN > 00111100000010010001001000110100
ASM > LUI $t1 0x1234

IN  < ORI $t1 $t1 0x5678 // t1=0x12345678

HEX > 0x35295678
BIN > 00110101001010010101011001111000
ASM > ORI $t1 $t1 0x5678

IN  < LUI $t2 0x0000

HEX > 0x3c0a0000
BIN > 00111100000010100000000000000000
ASM > LUI $t2 0x0

IN  < ORI $t2 $t2 0x0000 //t2=0x00000000

HEX > 0x354a0000
BIN > 00110101010010100000000000000000
ASM > ORI $t2 $t2 0x0

IN  < SW $t1 0x0000 $t2   //Store 12345678 in address 00000000

HEX > 0xad490000
BIN > 10101101010010010000000000000000
ASM > SW $t1 0x0 $t2

IN  < ANDI $t1 $t1 0x0000 //t1=0000
```

```
HEX > 0x31290000
BIN > 00110001001010010000000000000000
ASM > ANDI $t1 $t1 0x0

IN  < LUI $t1 0xABCD           //t1=ABCD0000

HEX > 0x3c09abcd
BIN > 00111100000010011010101111001101
ASM > LUI $t1 0xabcd

IN  < ORI $t1 $t1 0xEF12  //t1=ABCDEF12

HEX > 0x3529ef12
BIN > 00110101001010011110111100010010
ASM > ORI $t1 $t1 0xef12

IN  < SW $t1 0x0004 $t2   //Store ABCDEF12 in address 00000004

HEX > 0xad490004
BIN > 10101101010010010000000000000100
ASM > SW $t1 0x4 $t2

IN  < ANDI $t1 $t1 0x0000

HEX > 0x31290000
BIN > 00110001001010010000000000000000
ASM > ANDI $t1 $t1 0x0

IN  < LUI $t1 0x1122

HEX > 0x3c091122
BIN > 00111100000010010001000100100010
ASM > LUI $t1 0x1122

IN  < ORI $t1 $t1 0x3344  // t1=11223344

HEX > 0x35293344
BIN > 00110101001010010011001101000100
ASM > ORI $t1 $t1 0x3344

IN  < SW $t1 0x0008 $t2   //Store 11223344 in address 00000008

HEX > 0xad490008
BIN > 10101101010010010000000000001000
ASM > SW $t1 0x8 $t2

IN  < BEQ $t5 $t6 0x0003

HEX > 0x11ae0003
BIN > 00010001101011100000000000000011
ASM > BEQ $t5 $t6 0x3

IN  < ORI $t7 $t7 0xFFFF  // t7=0xffff

HEX > 0x35efffff
BIN > 00110101111011111111111111111111
ASM > ORI $t7 $t7 0xffff

IN  < ORI $t0 $t0 0xDAAD  //t0=0xdaad

HEX > 0x3508daad
BIN > 00110101000010001101101010101101
ASM > ORI $t0 $t0 0xdaad
```

```
IN  < ORI $t0 $t0 0xEAAE  //t0=oxfaaf

HEX > 0x3508eaae
BIN > 00110101000010001110101010101110
ASM > ORI $t0 $t0 0xeaae

IN  < LW $t3 0x0000 $t2   //t3=12345678 //Skips here

HEX > 0x8d4b0000
BIN > 10001101010010110000000000000000
ASM > LW $t3 0x0 $t2

IN  < LW $t4 0x0004 $t2   //t4=ABCDEF12

HEX > 0x8d4c0004
BIN > 10001101010011000000000000000100
ASM > LW $t4 0x4 $t2

IN  < LW $t5 0x0008 $t2    //t5=11223344

HEX > 0x8d4d0008
BIN > 10001101010011010000000000001000
ASM > LW $t5 0x8 $t2

IN  < ADD $t6 $t3 $t4      //t6=BE02458A

HEX > 0x16c7020
BIN > 00000001011011000111000000100000
ASM > ADD $t6 $t3 $t4

IN  < SW $t6 0x000C $t2    //sTORE BE02458A in address 0000000c

HEX > 0xad4e000c
BIN > 10101101010011100000000000001100
ASM > SW $t6 0xc $t2

IN  < ADD $t0 $t4 $t5      //t0=bcf02256

HEX > 0x18d4020
BIN > 00000001100011010100000000100000
ASM > ADD $t0 $t4 $t5

IN  < SUB $t7 $t0 $t4      //t7=11223344

HEX > 0x10c7822
BIN > 00000001000011000111100000100010
ASM > SUB $t7 $t0 $t4

IN  < ADD $t1 $t7 $t0      //t1=ce12559a

HEX > 0x1e84820
BIN > 00000001111010000100100000100000
ASM > ADD $t1 $t7 $t0

IN  < ADD $t2 $Zero, $Zero

HEX > 0x5020
BIN > 00000000000000000101000000100000
ASM > ADD $t2 $zero $zero

IN  < ADD $t3 $t2 $t2

HEX > 0x14a5820
BIN > 00000001010010100101100000100000
```

```
ASM > ADD $t3 $t2 $t2

IN  < ORI $t2 $t2 0xAA76

HEX > 0x354aaa76
BIN > 00110101010010101010101001110110
ASM > ORI $t2 $t2 0xaa76

IN  < OR  $t1 $t1 $t2

HEX > 0x12a4825
BIN > 00000001001010100100100000100101
ASM > OR $t1 $t1 $t2

IN  < ADD $t2 $t3 $t3

HEX > 0x16b5020
BIN > 00000001011010110101000000100000
ASM > ADD $t2 $t3 $t3

IN  < ORI $t2 $t2 0x5

HEX > 0x354a0005
BIN > 00110101010010100000000000000101
ASM > ORI $t2 $t2 0x5

IN  < LUI $t2 0x0000A

HEX > 0x3c0a000a
BIN > 00111100000010100000000000001010
ASM > LUI $t2 0xa

IN  < ADD $t3 $t2 $Zero

HEX > 0x1405820
BIN > 00000001010000000101100000100000
ASM > ADD $t3 $t2 $zero

IN  < MUL $t0 $t2 $t3

HEX > 0x14b4018
BIN > 00000001010010110100000000011000
ASM > MUL $t0 $t2 $t3

Code read and assemble completed, total 34 instructions
Select mode: 1 - Instruction, 2 - Cycle, 3 - Burst: 1
How many lines of code do you want to run? 0 for all of them: 0
Running in Instruction mode
Have run: LUI $t1 0x1234
====================================
Program Counter: 3
Control Stall: 0
Clock Cycle: 5
Efficiency: 32%
------------------IF-ID---------------
Ins: LUI $t2 0x0
NPC: 2
------------------ID-EX---------------
Ins: ORI $t1 $t1 0x5678
NPC: 1
A: 0x12340000
B: 0x0
I: 0x5678
-----------------EX-MEM---------------
```

```
BUBBLE
----------------MEM-WB--------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x12340000
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ORI $t1 $t1 0x5678
====================================
Program Counter: 4
Control Stall: 0
Clock Cycle: 8
Efficiency: 37%
------------------IF-ID--------------
Ins: ORI $t2 $t2 0x0
NPC: 3
------------------ID-EX--------------
BUBBLE
-----------------EX-MEM--------------
BUBBLE
----------------MEM-WB--------------
Ins: LUI $t2 0x0
ALU: 0x0
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
```

```
$t0: 0x0
$t1: 0x12345678
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: LUI $t2 0x0
====================================
Program Counter: 5
Control Stall: 0
Clock Cycle: 9
Efficiency: 40%
------------------IF-ID---------------
Ins: SW $t1 0x0 $t2
NPC: 4
------------------ID-EX---------------
Ins: ORI $t2 $t2 0x0
NPC: 3
A: 0x0
B: 0x0
I: 0x0
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x12345678
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
```

```
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ORI $t2 $t2 0x0
====================================
Program Counter: 6
Control Stall: 0
Clock Cycle: 12
Efficiency: 38%
------------------IF-ID---------------
Ins: ANDI $t1 $t1 0x0
NPC: 5
------------------ID-EX---------------
Ins: SW $t1 0x0 $t2
NPC: 4
A: 0x0
B: 0x12345678
I: 0x0
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x12345678
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
```

```
$fp: 0x0
$ra: 0x0
Have run: SW $t1 0x0 $t2
====================================
Program Counter: 7
Control Stall: 0
Clock Cycle: 15
Efficiency: 37%
------------------IF-ID---------------
Ins: LUI $t1 0xabcd
NPC: 6
------------------ID-EX---------------
Ins: ANDI $t1 $t1 0x0
NPC: 5
A: 0x12345678
B: 0x0
I: 0x0
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x12345678
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ANDI $t1 $t1 0x0
====================================
Program Counter: 8
Control Stall: 0
Clock Cycle: 18
Efficiency: 38%
------------------IF-ID---------------
Ins: ORI $t1 $t1 0xef12
NPC: 7
```

```
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
Ins: LUI $t1 0xabcd
ALU: 0xabcd0000
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x0
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: LUI $t1 0xabcd
====================================
Program Counter: 9
Control Stall: 0
Clock Cycle: 19
Efficiency: 40%
------------------IF-ID---------------
Ins: SW $t1 0x4 $t2
NPC: 8
------------------ID-EX---------------
Ins: ORI $t1 $t1 0xef12
NPC: 7
A: 0xabcd0000
B: 0x0
I: 0xef12
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
```

```
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0xabcd0000
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ORI $t1 $t1 0xef12
====================================
Program Counter: 10
Control Stall: 0
Clock Cycle: 22
Efficiency: 39%
------------------IF-ID---------------
Ins: ANDI $t1 $t1 0x0
NPC: 9
------------------ID-EX---------------
Ins: SW $t1 0x4 $t2
NPC: 8
A: 0x0
B: 0xabcdef12
I: 0x4
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0xabcdef12
$t2: 0x0
$t3: 0x0
```

```
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: SW $t1 0x4 $t2
====================================
Program Counter: 11
Control Stall: 0
Clock Cycle: 25
Efficiency: 38%
------------------IF-ID---------------
Ins: LUI $t1 0x1122
NPC: 10
------------------ID-EX---------------
Ins: ANDI $t1 $t1 0x0
NPC: 9
A: 0xabcdef12
B: 0x0
I: 0x0
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0xabcdef12
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
```

```
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ANDI $t1 $t1 0x0
====================================
Program Counter: 12
Control Stall: 0
Clock Cycle: 28
Efficiency: 39%
------------------IF-ID---------------
Ins: ORI $t1 $t1 0x3344
NPC: 11
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
BUBBLE
---------------MEM-WB---------------
Ins: LUI $t1 0x1122
ALU: 0x11220000
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x0
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: LUI $t1 0x1122
====================================
Program Counter: 13
Control Stall: 0
```

```
Clock Cycle: 29
Efficiency: 40%
------------------IF-ID---------------
Ins: SW $t1 0x8 $t2
NPC: 12
------------------ID-EX---------------
Ins: ORI $t1 $t1 0x3344
NPC: 11
A: 0x11220000
B: 0x0
I: 0x3344
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11220000
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ORI $t1 $t1 0x3344
====================================
Program Counter: 14
Control Stall: 0
Clock Cycle: 32
Efficiency: 39%
------------------IF-ID---------------
Ins: BEQ $t5 $t6 0x3
NPC: 13
------------------ID-EX---------------
Ins: SW $t1 0x8 $t2
NPC: 12
A: 0x0
B: 0x11223344
I: 0x8
```

```
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
BEQ have A0 B: 0
Have run: SW $t1 0x8 $t2
====================================
Program Counter: 18
Control Stall: 0
Clock Cycle: 35
Efficiency: 40%
-----------------IF-ID---------------
Ins: LW $t3 0x0 $t2
NPC: 17
-----------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
Ins: BEQ $t5 $t6 0x3
ALU: 0x0
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
```

```
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: BEQ $t5 $t6 0x3
====================================
Program Counter: 19
Control Stall: 0
Clock Cycle: 36
Efficiency: 40%
------------------IF-ID---------------
Ins: LW $t4 0x4 $t2
NPC: 18
------------------ID-EX---------------
Ins: LW $t3 0x0 $t2
NPC: 17
A: 0x0
B: 0x0
I: 0x0
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x0
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
```

```
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: LW $t3 0x0 $t2
====================================
Program Counter: 21
Control Stall: 0
Clock Cycle: 39
Efficiency: 42%
------------------IF-ID---------------
Ins: ADD $t6 $t3 $t4
NPC: 20
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
Ins: LW $t5 0x8 $t2
ALU: 0x8
COND: 0x1c
B: 0x0
----------------MEM-WB---------------
Ins: LW $t4 0x4 $t2
ALU: 0x4
LMD: 0xabcdef12
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x12345678
$t4: 0x0
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
```

```
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: LW $t4 0x4 $t2
====================================
Program Counter: 22
Control Stall: 0
Clock Cycle: 40
Efficiency: 43%
------------------IF-ID---------------
Ins: SW $t6 0xc $t2
NPC: 21
------------------ID-EX---------------
Ins: ADD $t6 $t3 $t4
NPC: 20
A: 0x12345678
B: 0xabcdef12
I: 0x7020
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
Ins: LW $t5 0x8 $t2
ALU: 0x8
LMD: 0x11223344
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x0
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: LW $t5 0x8 $t2
====================================
Program Counter: 22
Control Stall: 0
```

```
Clock Cycle: 41
Efficiency: 43%
------------------IF-ID---------------
Ins: SW $t6 0xc $t2
NPC: 21
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
Ins: ADD $t6 $t3 $t4
ALU: 0xbe02458a
COND: 0x7035
B: 0xabcdef12
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0x0
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ADD $t6 $t3 $t4
====================================
Program Counter: 23
Control Stall: 0
Clock Cycle: 43
Efficiency: 43%
------------------IF-ID---------------
Ins: ADD $t0 $t4 $t5
NPC: 22
------------------ID-EX---------------
Ins: SW $t6 0xc $t2
NPC: 21
A: 0x0
B: 0xbe02458a
I: 0xc
-----------------EX-MEM---------------
```

```
BUBBLE
----------------MEM-WB--------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: SW $t6 0xc $t2
====================================
Program Counter: 24
Control Stall: 0
Clock Cycle: 46
Efficiency: 43%
------------------IF-ID--------------
Ins: SUB $t7 $t0 $t4
NPC: 23
------------------ID-EX--------------
BUBBLE
-----------------EX-MEM--------------
BUBBLE
----------------MEM-WB--------------
Ins: ADD $t0 $t4 $t5
ALU: 0xbcf02256
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
```

```
$t0: 0x0
$t1: 0x11223344
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ADD $t0 $t4 $t5
====================================
Program Counter: 25
Control Stall: 0
Clock Cycle: 47
Efficiency: 43%
------------------IF-ID---------------
Ins: ADD $t1 $t7 $t0
NPC: 24
------------------ID-EX---------------
Ins: SUB $t7 $t0 $t4
NPC: 23
A: 0xbcf02256
B: 0xabcdef12
I: 0x7822
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0x11223344
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x0
$s0: 0x0
$s1: 0x0
$s2: 0x0
```

```
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: SUB $t7 $t0 $t4
====================================
Program Counter: 26
Control Stall: 0
Clock Cycle: 50
Efficiency: 43%
------------------IF-ID---------------
Ins: ADD $t2 $zero $zero
NPC: 25
------------------ID-EX---------------
Ins: ADD $t1 $t7 $t0
NPC: 24
A: 0x11223344
B: 0xbcf02256
I: 0x4820
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0x11223344
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
```

```
$fp: 0x0
$ra: 0x0
Have run: ADD $t1 $t7 $t0
====================================
Program Counter: 27
Control Stall: 0
Clock Cycle: 53
Efficiency: 43%
------------------IF-ID---------------
Ins: ADD $t3 $t2 $t2
NPC: 26
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
Ins: ADD $t2 $zero $zero
ALU: 0x0
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12559a
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ADD $t2 $zero $zero
====================================
Program Counter: 28
Control Stall: 0
Clock Cycle: 54
Efficiency: 43%
------------------IF-ID---------------
Ins: ORI $t2 $t2 0xaa76
NPC: 27
------------------ID-EX---------------
Ins: ADD $t3 $t2 $t2
```

```
NPC: 26
A: 0x0
B: 0x0
I: 0x5820
-----------------EX-MEM--------------
BUBBLE
----------------MEM-WB--------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12559a
$t2: 0x0
$t3: 0x12345678
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ADD $t3 $t2 $t2
====================================
Program Counter: 29
Control Stall: 0
Clock Cycle: 57
Efficiency: 43%
-----------------IF-ID--------------
Ins: OR $t1 $t1 $t2
NPC: 28
-----------------ID-EX--------------
BUBBLE
-----------------EX-MEM--------------
BUBBLE
----------------MEM-WB--------------
Ins: ORI $t2 $t2 0xaa76
ALU: 0xaa76
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
```

```
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12559a
$t2: 0x0
$t3: 0x0
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ORI $t2 $t2 0xaa76
====================================
Program Counter: 30
Control Stall: 0
Clock Cycle: 58
Efficiency: 44%
------------------IF-ID---------------
Ins: ADD $t2 $t3 $t3
NPC: 29
------------------ID-EX---------------
Ins: OR $t1 $t1 $t2
NPC: 28
A: 0xce12559a
B: 0xaa76
I: 0x4825
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12559a
$t2: 0xaa76
$t3: 0x0
$t4: 0xabcdef12
$t5: 0x11223344
```

```
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: OR $t1 $t1 $t2
====================================
Program Counter: 31
Control Stall: 0
Clock Cycle: 61
Efficiency: 44%
------------------IF-ID---------------
Ins: ORI $t2 $t2 0x5
NPC: 30
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
Ins: ADD $t2 $t3 $t3
ALU: 0x0
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12fffe
$t2: 0xaa76
$t3: 0x0
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
```

```
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ADD $t2 $t3 $t3
====================================
Program Counter: 32
Control Stall: 0
Clock Cycle: 62
Efficiency: 44%
------------------IF-ID---------------
Ins: LUI $t2 0xa
NPC: 31
------------------ID-EX---------------
Ins: ORI $t2 $t2 0x5
NPC: 30
A: 0x0
B: 0x0
I: 0x5
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12fffe
$t2: 0x0
$t3: 0x0
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ORI $t2 $t2 0x5
====================================
Program Counter: 33
Control Stall: 0
Clock Cycle: 65
Efficiency: 44%
```

```
------------------IF-ID---------------
Ins: ADD $t3 $t2 $zero
NPC: 32
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
Ins: LUI $t2 0xa
ALU: 0xa0000
LMD: 0x0
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12fffe
$t2: 0x5
$t3: 0x0
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: LUI $t2 0xa
====================================
Program Counter: 34
Control Stall: 0
Clock Cycle: 66
Efficiency: 44%
------------------IF-ID---------------
Ins: MUL $t0 $t2 $t3
NPC: 33
------------------ID-EX---------------
Ins: ADD $t3 $t2 $zero
NPC: 32
A: 0xa0000
B: 0x0
I: 0x5820
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
```

```
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12fffe
$t2: 0xa0000
$t3: 0x0
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: ADD $t3 $t2 $zero
====================================
Program Counter: 34
Control Stall: 0
Clock Cycle: 69
Efficiency: 44%
------------------IF-ID---------------
BUBBLE
------------------ID-EX---------------
Ins: MUL $t0 $t2 $t3
NPC: 33
A: 0xa0000
B: 0xa0000
I: 0x4018
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0xbcf02256
$t1: 0xce12fffe
```

```
$t2: 0xa0000
$t3: 0xa0000
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
Have run: MUL $t0 $t2 $t3
====================================
Program Counter: 34
Control Stall: 0
Clock Cycle: 72
Efficiency: 43%
------------------IF-ID---------------
BUBBLE
------------------ID-EX---------------
BUBBLE
-----------------EX-MEM---------------
BUBBLE
----------------MEM-WB---------------
BUBBLE
====================================
'R' for registers and continue, anything else to continue...r
$zero: 0x0
$at: 0x0
$v0: 0x0
$v1: 0x0
$a0: 0x0
$a1: 0x0
$a2: 0x0
$a3: 0x0
$t0: 0x0
$t1: 0x64
$t2: 0xa0000
$t3: 0xa0000
$t4: 0xabcdef12
$t5: 0x11223344
$t6: 0xbe02458a
$t7: 0x11223344
$s0: 0x0
$s1: 0x0
$s2: 0x0
$s3: 0x0
$s4: 0x0
$s5: 0x0
$s6: 0x0
$s7: 0x0
$t8: 0x0
$t9: 0x0
```

```
$k0: 0x0
$k1: 0x0
$gp: 0x0
$sp: 0x0
$fp: 0x0
$ra: 0x0
MEMORY DUMP:
0: 12345678                1: 0                2: 0                3: 0
       4: abcdef12                5: 0                6: 0                7: 0

8: 11223344                9: 0                10: 0               11: 0
       12: be02458a               13: 0               14: 0               15: 0

16: 0               17: 0               18: 0               19: 0               20:
0                   21: 0               22: 0               23: 0
24: 0               25: 0               26: 0               27: 0               28:
0                   29: 0               30: 0               31: 0
32: 0               33: 0               34: 0               35: 0               36:
0                   37: 0               38: 0               39: 0
40: 0               41: 0               42: 0               43: 0               44:
0                   45: 0               46: 0               47: 0
48: 0               49: 0               50: 0               51: 0               52:
0                   53: 0               54: 0               55: 0
56: 0               57: 0               58: 0               59: 0               60:
0                   61: 0               62: 0               63: 0
64: 0               65: 0               66: 0               67: 0               68:
0                   69: 0               70: 0               71: 0
72: 0               73: 0               74: 0               75: 0               76:
0                   77: 0               78: 0               79: 0
80: 0               81: 0               82: 0               83: 0               84:
0                   85: 0               86: 0               87: 0
88: 0               89: 0               90: 0               91: 0               92:
0                   93: 0               94: 0               95: 0
96: 0               97: 0               98: 0               99: 0
       100: 0              101: 0              102: 0              103: 0

104: 0              105: 0              106: 0              107: 0
       108: 0              109: 0              110: 0              111: 0

112: 0              113: 0              114: 0              115: 0
       116: 0              117: 0              118: 0              119: 0

120: 0              121: 0              122: 0              123: 0
       124: 0              125: 0              126: 0              127: 0

128: 0              129: 0              130: 0              131: 0
       132: 0              133: 0              134: 0              135: 0

136: 0              137: 0              138: 0              139: 0
       140: 0              141: 0              142: 0              143: 0

144: 0              145: 0              146: 0              147: 0
       148: 0              149: 0              150: 0              151: 0

152: 0              153: 0              154: 0              155: 0
       156: 0              157: 0              158: 0              159: 0

160: 0              161: 0              162: 0              163: 0
       164: 0              165: 0              166: 0              167: 0

168: 0              169: 0              170: 0              171: 0
       172: 0              173: 0              174: 0              175: 0
```

176: 0 177: 0 178: 0 179: 0
180: 0 181: 0 182: 0 183: 0

184: 0 185: 0 186: 0 187: 0
188: 0 189: 0 190: 0 191: 0

192: 0 193: 0 194: 0 195: 0
196: 0 197: 0 198: 0 199: 0

200: 0 201: 0 202: 0 203: 0
204: 0 205: 0 206: 0 207: 0

208: 0 209: 0 210: 0 211: 0
212: 0 213: 0 214: 0 215: 0

216: 0 217: 0 218: 0 219: 0
220: 0 221: 0 222: 0 223: 0

224: 0 225: 0 226: 0 227: 0
228: 0 229: 0 230: 0 231: 0

232: 0 233: 0 234: 0 235: 0
236: 0 237: 0 238: 0 239: 0

240: 0 241: 0 242: 0 243: 0
244: 0 245: 0 246: 0 247: 0

248: 0 249: 0 250: 0 251: 0
252: 0 253: 0 254: 0 255: 0

256: 0 257: 0 258: 0 259: 0
260: 0 261: 0 262: 0 263: 0

264: 0 265: 0 266: 0 267: 0
268: 0 269: 0 270: 0 271: 0

272: 0 273: 0 274: 0 275: 0
276: 0 277: 0 278: 0 279: 0

280: 0 281: 0 282: 0 283: 0
284: 0 285: 0 286: 0 287: 0

288: 0 289: 0 290: 0 291: 0
292: 0 293: 0 294: 0 295: 0

296: 0 297: 0 298: 0 299: 0
300: 0 301: 0 302: 0 303: 0

304: 0 305: 0 306: 0 307: 0
308: 0 309: 0 310: 0 311: 0

312: 0 313: 0 314: 0 315: 0
316: 0 317: 0 318: 0 319: 0

320: 0 321: 0 322: 0 323: 0
324: 0 325: 0 326: 0 327: 0

328: 0 329: 0 330: 0 331: 0
332: 0 333: 0 334: 0 335: 0

336: 0 337: 0 338: 0 339: 0
340: 0 341: 0 342: 0 343: 0

344: 0 345: 0 346: 0 347: 0
348: 0 349: 0 350: 0 351: 0

352: 0 353: 0 354: 0 355: 0
356: 0 357: 0 358: 0 359: 0

360: 0 361: 0 362: 0 363: 0
364: 0 365: 0 366: 0 367: 0

368: 0 369: 0 370: 0 371: 0
372: 0 373: 0 374: 0 375: 0

376: 0 377: 0 378: 0 379: 0
380: 0 381: 0 382: 0 383: 0

384: 0 385: 0 386: 0 387: 0
388: 0 389: 0 390: 0 391: 0

392: 0 393: 0 394: 0 395: 0
396: 0 397: 0 398: 0 399: 0

400: 0 401: 0 402: 0 403: 0
404: 0 405: 0 406: 0 407: 0

408: 0 409: 0 410: 0 411: 0
412: 0 413: 0 414: 0 415: 0

416: 0 417: 0 418: 0 419: 0
420: 0 421: 0 422: 0 423: 0

424: 0 425: 0 426: 0 427: 0
428: 0 429: 0 430: 0 431: 0

432: 0 433: 0 434: 0 435: 0
436: 0 437: 0 438: 0 439: 0

440: 0 441: 0 442: 0 443: 0
444: 0 445: 0 446: 0 447: 0

448: 0 449: 0 450: 0 451: 0
452: 0 453: 0 454: 0 455: 0

456: 0 457: 0 458: 0 459: 0
460: 0 461: 0 462: 0 463: 0

464: 0 465: 0 466: 0 467: 0
468: 0 469: 0 470: 0 471: 0

472: 0 473: 0 474: 0 475: 0
476: 0 477: 0 478: 0 479: 0

480: 0 481: 0 482: 0 483: 0
484: 0 485: 0 486: 0 487: 0

488: 0 489: 0 490: 0 491: 0
492: 0 493: 0 494: 0 495: 0

496: 0 497: 0 498: 0 499: 0
500: 0 501: 0 502: 0 503: 0

504: 0 505: 0 506: 0 507: 0
508: 0 509: 0 510: 0 511: 0

```
All available instructions have been run
Do you wish to run another round of this simulation? yN: r
Good bye!

Process finished with exit code 0
```