Report

In this project, I created a simple pipeline CPU that can solve with hazards.

To implement the CPU, you should go to the **HCPU.v** and change the string in the 52^{nd} line into the **ABSOLUTE PATH** of the test file, and set the period in **test.v** to an adequate value.

```
52        datafile = $fopen ("C:\\Users\\king\\Desktop\\file2.txt", "r");
```

Then, type as follows: (you should change the address to the **ABSOLUTE PATH** of the file as well)

```
C:\Users\king>iverilog -o CPU C:\Users\king\Desktop\HCPU.V C:\Users\king\Desktop\test.V

C:\Users\king>vvp CPU
```

Then it will run automatically.

Initialize

Firstly, the general registers will be set to 0. The gr[0] is $zero, meaning that once it was set to a non-zero value, it will be set back to zero.

Then, the program will start to read the test file and store the instructions in the instruction memory. **The maximum is 64 instructions**, but could be modified by changing the “instruction” register.

Pipeline

The CPU process could be divided into 7 parts: Load, Instruction fetch, Instruction Decode, Execute, Memory interact, Write back, Reload. Although in sequence, those parts are mutually independent.

Load

As the first stage of the pipeline, the 5 main stages (IF - WB) will be assigned corresponding data. The main stages are parallel (e.g. the IF stage will be assigned 5^{th} instruction will the WB stage will be assigned 1^{st} instruction).

Instruction Fetch

In this stage, the program will extract instructions from the instruction memory controlled by the pc value. The instruction will be assigned to the ID part in the load and reload stages.

Instruction Decode

In this stage, the program will decode the given instruction. The elements are as follows:

Name	Function	Name	Function
opcode	The op part of instruction	functcode	The func part of instruction
immcode	The imm part of instruction	imm	Sign/zero-extended immediate number
SignExt	1:Sign-extension of imm part 0:Zero-extension of imm part	stall&cont	Store if the machine should stall in the next clock
reg_A/B	The registers for execution	reg_Dest	The destination of data
a/bAbsent	1: reg_A/B will be assigned in reload part	ALUop	The operation of coming EXE part
Branch jump	Store the coming change of PC	FlagExamine	Store the flags that should be examined in EXE part

s/l	Store if interaction with memory is needed	neg	Store if the negative flag should be examined in EXE part

The ALUop reads as follows:

ALUop	Operator	ALUop	Operator	ALUop	Operator	ALUop	Operator
0	+	1	-	2	&	3	|
4	^	5	~|(nor)	6	<<	7	>>/>>>

The FlagExamine & neg reads as follows:

FlagExamine[2]	FlagExamine[1]	FlagExamine[0]	neg
The Unsign bit: if set to 0, 1. The overflow flag will not be examined (e.g. addu) 2. The right-shift operation will be arithmetic (e.g. sra)	If set to 1, the zero flag will be examined (e.g. beq, bne)	If the zero flag is found equal to this bit, the branch operation will be operated.	If set to 1, the negative flag should be examined (e.g. slt)

The Branchjump reads as follows:

00	01	10	11
PC does not change	Branch	Jump	Jump-and-link

The reg_A and reg_B will be assigned an adequate value, most frequently the value in corresponding registers.

If	Reg_A=	Reg_B=
Op = 0 and func = 0000xx	shamt	rt
Op = 0000xx	2	Instruction [25:0]

Except the stall & cont is global variable (will be mentioned in the hazard part), all other data will be sent into the buffer between ID part and EXE part.

Execute

The EXE part will first calculate reg_A and reg_B by given operators and get the result. Then the flags will start to be examined. This part is mainly implemented in Project 3.

Branch in pipeline

The new pc after branching will be calculated from the old pc in IF part, minus 8 (2 stages difference between IF and EXE), then add the branch.

Memory

The MEM part will interact with memory. If the s was set to 1 when the instruction was in the ID part, the value in the destination register will be saved in the specific address indicated by the result. If the l was set to 1, the value in the specific address will be saved in the destination register. In this program, the size of memory is 32*256 bits.

Write Back

The WB part will write results back to the destination register. If the instruction is sw (save = 1) or j/beq/ne (branchjump = 10/01, i.e. branchjump[1]^branchjump[0] = 1), the WB part will be skipped.

Reload

The reload part will gather the critical data in those parts and send them into the buffer between the part and its next part. So at the load part at the next clock, the data could be loaded to the next part, and a pipeline is created.

Hazards

There are 2 main hazards: data hazard and control hazard.

The following elements will be appended to solve hazards:

Stall & cont	Stall the IF/ID part	registers	Store the destination registers in pipeline
a/bAbsent	Mark the data hazard	flush	Clear the IF/ID part

The “registers” will store the destination registers in the pipeline. Every time an instruction is decoded, the register will right shift and load the destination register. If the reg_A/B is in conflict with one of the reg_D, the program will take the following measures to prevent hazards:

Conflicts	Meaning	Measure
instr[25:21] == 0 instr[20:16] == 0	No Conflicts	Reg_A = 0 Reg_B = 0
instr[25:21] == registers[14:10] instr[20:16] == registers[14:10]	The conflict data is in the EXE part	Stall = 1 Cont = 1
instr[25:21] == registers[9:5] instr[20:16] == registers[9:5]	The conflict data is in the MEM part	Load was 1: set a/bAbsent to 1 Load was 0: set to the result from MEM part
instr[25:21] == registers[4:0] instr[20:16] == registers[4:0]	The conflict data is in the WB part	set to the result from WB part

Some instructions (e.g. jr right after j) will call a stall even there is no hazard.

If a/bAbsent is set to 1, the reg_A/B will be set to the result sending to the buffer between MEM and WB at the reloading stage. This can be done at the negedge of clock in reality.

When stall is set to 1, it means that the IF & ID part will be stalled for 2 cycles. This includes:

Stall the reloading of IF and ID
Modify the “registers”

Stall the change of pc (while cont will be set to 0)
Stall the execution of IF and ID part
Stall the reloading of IF and ID
Modify the “registers”
Stall the change of pc (while stall will be set to 0)

In case that the IF and ID will be overwritten before executed, the stall cycle will be 2.
Due to flaws in the modification of registers, **IT IS STRONGLY ADVISED THAT THE $31 SHOULD NOT BE USED IN TEST FILE IN CASE OF INDEFINITE STALL.**

Meanwhile, if branch and jump operations occurs, the flush bit will be set to 1.
In the reloading stage, if the flush bit is 1, everything of IF and ID stage will be cleared, and in the next clock the EXE part will handle with a nop (sll $zero $zero 0).

Test files
There are two attached test files: file.txt and file2.txt.
In file.txt, the following instructions are used:
addi add slt ori beq jal sw sub lw jr
In file2.txt, the following instructions are used:
addi sll sllv sra nor j addu

The results are as follows:

r	Op	Rs	Rt	Rd	Shamt	Func
i	Op	Rs	Rt	Imm		
j	Op	New pc				

Instruction	ALUop	Reg_A	Reg_B	Reg_Dest	FlagExamine	Neg	branchjump
Addi	0(+)	Rs	Imm	Rt	000	0	00
Add	0(+)	Rs	Rt	Rd	000	0	00
Slt	1(-)	Rs	Rt	Rd	000	1	00
Ori	3(|)	Rs	Imm	Rt	000	0	00
Beq	1(-)	Rs	Rt	30[1]	010	0	01
Jal	6(<<)[2]	2	New pc	30	000	0	11
Sw	0(+)	Rs	Imm	Rt[3]	000	0	00
Sub	1(-)	Rs	Rt	Rd	000	0	00
Lw	0(+)	Rs	Imm	Rt	000	0	00
Jr	0(+)	Rs	Rt(0)	Rd(0)	000	0	10
Sll	6(<<)	Shamt	Rt	Rd	000	0	00
Sllv	6(<<)	Rs	Rt	Rd	000	0	00
Sra	7(>>)	Rs	Rt	Rd	100[4]	0	00

1 The reg_Dest in blue will not be really implemented.
2 The << and >> operation will be reg_B <<(>>) reg_A.
3 The data in rt will be stored in specific address.
4 The “Unsign” flag indicate the arithmetic operation.

Nor	5(nor)	Rs	Rt	Rd	000	0	00
J	6(<<)	2	New pc	30	000	0	10
Addu	0(+)	Rs	Rt	Rd	100	0	00

File.txt

```
Addi $1, $0, 0xf
Addi $2, $0, 0x11
Addi $4, $4, 0x40
Add $3, $1, $2
Add $1, $3, $2
Slt $5, $4, $3
Ori $6, $8, 0x1
Beq $6, $5, 0x1
Jal 0x3
Addi $7, $0, 0x1
Sw $3, 1($7)
Sub $3, $3, $5
Lw $6, 1($7)
Beq $3, $4, 1
Jr $30
```

File2.txt

```
Addi $1, $0, 1
Sllv $1, $1, $1
Sll $1, $1, 31
Sra $1, $1, 24
Nor $1, $0, $0
Sll $1, $1, 31
Addu $1, $1, $1
J 0
```

File.txt

```
pc  :         instruction             :  gr1    :  gr2    :  gr3    :  gr4    :  gr5    :  gr6    :  gr7    :  gr30   :   ADD
xxxx:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0000:00000000000000000000000000000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0004:00100000000000010000000000001111:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00100000000000100000000000010001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00100000100001000000000001000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000001000100001100000100000:0000000f:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0014:00000000011000100000100000100000:0000000f:00000011:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0014:00000000011000100000100000100000:0000000f:00000011:00000000:00000040:00000000:00000000:00000000:00000000:xxxxxxxx
0014:00000000011000100000100000100000:0000000f:00000011:00000020:00000040:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000100000110010100000101010:0000000f:00000011:00000020:00000040:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00110101000001100000000000000001:0000000f:00000011:00000020:00000040:00000000:00000000:00000000:00000000:xxxxxxxx
0020:00010000110001010000000000000001:00000031:00000011:00000020:00000040:00000000:00000000:00000000:00000000:xxxxxxxx
0020:00010000110001010000000000000001:00000031:00000011:00000020:00000040:00000000:00000000:00000000:00000000:xxxxxxxx
0024:00001100000000000000000000000011:00000031:00000011:00000020:00000040:00000000:00000001:00000000:00000000:xxxxxxxx
0028:00100000000111100000000000000001:00000031:00000011:00000020:00000040:00000000:00000001:00000000:00000000:xxxxxxxx
000c:00000000000000000000000000000000:00000031:00000011:00000020:00000040:00000000:00000001:00000000:00000000:xxxxxxxx
0010:00000000001000100001100000100000:00000031:00000011:00000020:00000040:00000000:00000001:00000000:00000028:xxxxxxxx
0014:00000000011000100000100000100000:00000031:00000011:00000020:00000040:00000000:00000001:00000000:00000028:xxxxxxxx
0014:00000000011000100000100000100000:00000031:00000011:00000042:00000040:00000000:00000001:00000000:00000028:xxxxxxxx
0018:00000000100000110010100000101010:00000031:00000011:00000042:00000040:00000000:00000001:00000000:00000028:xxxxxxxx
001c:00110101000001100000000000000001:00000031:00000011:00000042:00000040:00000000:00000001:00000000:00000028:xxxxxxxx
0020:00010000110001010000000000000001:00000053:00000011:00000042:00000040:00000000:00000001:00000000:00000028:xxxxxxxx
0020:00010000110001010000000000000001:00000053:00000011:00000042:00000040:00000001:00000001:00000000:00000028:xxxxxxxx
0024:00001100000000000000000000000011:00000053:00000011:00000042:00000040:00000001:00000001:00000000:00000028:xxxxxxxx
0024:00000000000000000000000000000000:00000053:00000011:00000042:00000040:00000001:00000001:00000000:00000028:xxxxxxxx
0028:00100000000111100000000000000001:00000053:00000011:00000042:00000040:00000001:00000001:00000000:00000028:xxxxxxxx
002c:10101100111000110000000000000001:00000053:00000011:00000042:00000040:00000001:00000001:00000000:00000028:xxxxxxxx
0030:00000000011001010001100000100010:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:xxxxxxxx
0030:00000000011001010001100000100010:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:00000042
0030:00000000011001010001100000100010:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:00000042
0034:10001100111001100000000000000001:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:00000042
0038:00010000011001000000000000000001:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:00000042
003c:00000011110000000000000000001000:00000053:00000011:00000041:00000040:00000001:00000001:00000001:00000028:00000042
003c:00000011110000000000000000001000:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0040:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0028:00000000000000000000000000000000:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
002c:10101100111000110000000000000001:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0030:00000000011001010001100000100010:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0030:00000000011001010001100000100010:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000041
```

```
0030:00000000011001010001100000100010:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:00000042
0034:10001100111001100000000000000001:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:00000042
0038:00010000011001000000000000000001:00000053:00000011:00000042:00000040:00000001:00000001:00000001:00000028:00000042
003c:00000011110000000000000000001000:00000053:00000011:00000041:00000040:00000001:00000001:00000001:00000028:00000042
003c:00000011110000000000000000001000:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0040:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0028:00000000000000000000000000000000:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
002c:10101100111000110000000000000001:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0030:00000000011001010001100000100010:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000042
0030:00000000011001010001100000100010:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000041
0030:00000000011001010001100000100010:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000041
0034:10001100111001100000000000000001:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000041
0038:00010000011001000000000000000001:00000053:00000011:00000041:00000040:00000001:00000042:00000001:00000028:00000041
003c:00000011110000000000000000001000:00000053:00000011:00000040:00000040:00000001:00000042:00000001:00000028:00000041
003c:00000000000000000000000000000000:00000053:00000011:00000040:00000040:00000001:00000041:00000001:00000028:00000041
0040:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:00000053:00000011:00000040:00000040:00000001:00000041:00000001:00000028:00000041
```

File2.txt

```
0000:00000000000000000000000000000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0004:00100000000000001000000000000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00000000010000100001000000000100:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00000000010000100001000000000100:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0014:00000000000000000000100000100111:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0020:00001000000000000000000000000000:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0024:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0000:00000000000000000000000000000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0004:00100000000000001000000000000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00000000010000100001000000000100:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00000000010000100001000000000100:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0014:00000000000000000000100000100111:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0020:00001000000000000000000000000000:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0024:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0000:00000000000000000000000000000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0004:00100000000000001000000000000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00000000010000100001000000000100:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
```

```
0008:00000000010000100001000000000100:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0014:00000000000000000000100000100111:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100001:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0020:00001000000000000000000000000000:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0024:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
```

(The instructions are corresponding to the pc at the left minus 4; Meanwhile, the nop is due to flushing in the branch/jump process; there is no nops in the test files.)

If the addu in file2.txt is changed to add, it will act like:

```
pc  :          instruction             :  gr1   :  gr2   :  gr3   :  gr4   :  gr5   :  gr6   :  gr7   :  gr30  :   ADD
xxxx:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0000:00000000000000000000000000000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0004:00100000000000001000000000000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00000000010000100001000000000100:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0008:00000000010000100001000000000100:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000001:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
000c:00000000000000001000011111000000:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000002:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0010:00000000000000001000011100000011:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0014:00000000000000000000100000100111:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
0018:00000000000000001000011111000000:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100000:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100000:ffffffff:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
001c:00000000010000100001000000100000:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
terminated after the following exception: overflow detected in +/- operation
0020:00001000000000000000000000000000:80000000:00000000:00000000:00000000:00000000:00000000:00000000:00000000:xxxxxxxx
```