Code Modification:

1. All the modified code has comment with //Code Modified.
2. Insert the basic operation ADD and OR's opcode:
   ADD = 6'b100000, OR = 6'b100101.
3. IMemory[0] = 32'h8c210003;
   IMemory[1] = 32'hac020000;
   IMemory[2] = 32'h00642820;
   This code stand for:
   lw $1 3($1); //  IMemory[0] = 32'h8c210003;
   sw $2 0($0); // IMemory[1] = 32'hac020000;
   Add $3 $2 $5; // IMemory[2] = 32'h00642820;
   This is not our target code, so we need to change it to what we want.
   add $5,$2,$1 -> 32'b000000|00010|00001|00101|00000|100000
   IMemory[0] = 32'h00412820

   lw $3,4($5) -> 32'b100011|00101|00011|0000000000000100
   IMemory[1] = 32'h8ca30004;

   lw $2,0($2) -> 32'b100011|00010|00010|0000000000000000
   IMemory[2] = 32'h8c420000;

   or $3,$5,$3 -> 32'b000000|00101|00011|00011|00000|100101
   IMemory[3] = 32'h00a31825;

   sw $3,0($5) -> 32'b101011|00101|00011|000000000000000
   IMemory[4] = 32'haca30000;
   Then we got the none-nop code: (This is in the mipspipe_harzard.v)
   IMemory[0] = 32'h00412820;//add $5,$2,$1
   IMemory[1] = 32'h8ca30004;//lw $3,4($5)
   IMemory[2] = 32'h8c420000;//lw $2,0($2)
   IMemory[3] = 32'h00a31825;//or $3,$5,$3
   IMemory[4] = 32'haca30000;//sw $3,0($5)

4. After we got the code, we can see IMemory[0] and [1] has the data dependency, so add two nops instruction between them, also [1] and [3] has data dependency, since they have [2] between them, so only one nop will be needed. Same between [3] and [4]. After optimized, the code will be look like this:

   IMemory[0] = 32'h00412820;//add $5,$2,$1
   IMemory[1] = nop;//insert bubble
   IMemory[2] = nop;//insert bubble
   IMemory[3] = 32'h8ca30004;//lw $3,4($5)
   IMemory[4] = 32'h8c420000;//lw $2,0($2)
   IMemory[5] = nop;//lw $2,0($2) has taken 1 lie, so only need 1 bubble here
   IMemory[6] = 32'h00a31825;//or $3,$5,$3

IMemory[7] = nop;// insert bubble
IMemory[8] = nop;// insert bubble
IMemory[9] = 32'haca30000;//sw $3,0($5)

5. Rewrite the DMemory, since will need to access those addresses:
DMemory[2] = 32'hffffffff0; //0($2):32'hffffffff0
DMemory[5] = 32'hffffffff; //4($5):32'hffffffff

6. We have opcode of add and or, but also need to implement the operation of add and or:
37: EXMEMALUOut <= Ain | Bin; // or operation

Comparison:

With Nop output:

clock cycle = 1 (time = 10)
IF/ID registers
IF/ID.PC+4 = 00000004, IF/ID.IR = 00412820

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = xxxxxxxx, EX/MEM.ALUout = xxxxxxxx
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = xxxxxxxx
EX/MEM.op = 00

clock cycle = 2 (time = 20)
IF/ID registers
IF/ID.PC+4 = 00000008, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 2, ID/EX.rt = 1
ID/EX.A = 00000002, ID/EX.B = 00000001
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 2, EX/MEM.rt = 1
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = xxxxxxxx
EX/MEM.op = 00

clock cycle = 3 (time = 30)
IF/ID registers
IF/ID.PC+4 = 0000000c, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000003, EX/MEM.ALUout = 00000001
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 4 (time = 40)
IF/ID registers
IF/ID.PC+4 = 00000010, IF/ID.IR = 8ca30004

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers

EX/MEM.rs =  0, EX/MEM.rt =  0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd =  5, MEM/WB.rt =  1
MEM/WB.value = 00000003
EX/MEM.op = 00

clock cycle =  5 (time = 50)
IF/ID registers
IF/ID.PC+4 = 00000014, IF/ID.IR = 8c420000

ID/EX registers
ID/EX.rs =  5, ID/EX.rt =  3
ID/EX.A = 00000005, ID/EX.B = 00000003
ID/EX.op = 23

EX/MEM registers
EX/MEM.rs =  5, EX/MEM.rt =  3
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd =  0, MEM/WB.rt =  0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle =  6 (time = 60)
IF/ID registers
IF/ID.PC+4 = 00000018, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs =  2, ID/EX.rt =  2
ID/EX.A = 00000002, ID/EX.B = 00000002
ID/EX.op = 23

EX/MEM registers
EX/MEM.rs =  2, EX/MEM.rt =  2

EX/MEM.ALUOut = 00000009, EX/MEM.ALUout = 00000003
EX/MEM.op = 23

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 7 (time = 70)
IF/ID registers
IF/ID.PC+4 = 0000001c, IF/ID.IR = 00a31825

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000002, EX/MEM.ALUout = 00000002
EX/MEM.op = 23

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 3
MEM/WB.value = fffffff0
EX/MEM.op = 23

clock cycle = 8 (time = 80)
IF/ID registers
IF/ID.PC+4 = 00000020, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 5, ID/EX.rt = 3
ID/EX.A = 00000003, ID/EX.B = 00000003
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 5, EX/MEM.rt = 3
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000

```
        EX/MEM.op = 00

MEM/WB registers
        MEM/WB.rd =  0, MEM/WB.rt =  2
        MEM/WB.value = 00000000
        EX/MEM.op = 23



clock cycle =  9 (time = 90)
IF/ID registers
        IF/ID.PC+4 = 00000024, IF/ID.IR = 00000020

ID/EX registers
        ID/EX.rs =  0, ID/EX.rt =  0
        ID/EX.A = 00000000, ID/EX.B = 00000000
        ID/EX.op = 00

EX/MEM registers
        EX/MEM.rs =  0, EX/MEM.rt =  0
        EX/MEM.ALUOut = 00000003, EX/MEM.ALUout = 00000003
        EX/MEM.op = 00

MEM/WB registers
        MEM/WB.rd =  0, MEM/WB.rt =  0
        MEM/WB.value = 00000000
        EX/MEM.op = 00



clock cycle = 10 (time = 100)
IF/ID registers
        IF/ID.PC+4 = 00000028, IF/ID.IR = aca30000

ID/EX registers
        ID/EX.rs =  0, ID/EX.rt =  0
        ID/EX.A = 00000000, ID/EX.B = 00000000
        ID/EX.op = 00

EX/MEM registers
        EX/MEM.rs =  0, EX/MEM.rt =  0
        EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
        EX/MEM.op = 00
```

MEM/WB registers
MEM/WB.rd = 3, MEM/WB.rt = 3
MEM/WB.value = 00000003
EX/MEM.op = 00

clock cycle = 11 (time = 110)
IF/ID registers
IF/ID.PC+4 = 0000002c, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 5, ID/EX.rt = 3
ID/EX.A = 00000003, ID/EX.B = fffffff0
ID/EX.op = 2b

EX/MEM registers
EX/MEM.rs = 5, EX/MEM.rt = 3
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 12 (time = 120)
IF/ID registers
IF/ID.PC+4 = 00000030, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000003, EX/MEM.ALUout = fffffff0
EX/MEM.op = 2b

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 13 (time = 130)
IF/ID registers
IF/ID.PC+4 = 00000034, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 3
MEM/WB.value = 00000000
EX/MEM.op = 2b

clock cycle = 14 (time = 140)
IF/ID registers
IF/ID.PC+4 = 00000038, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers

MEM/WB.rd =  0, MEM/WB.rt =  0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 15 (time = 150)
IF/ID registers
IF/ID.PC+4 = 0000003c, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs =  0, ID/EX.rt =  0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs =  0, EX/MEM.rt =  0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd =  0, MEM/WB.rt =  0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle =  0 (time = 160)

Without Nop output:.
clock cycle =  1 (time = 10)
IF/ID registers
IF/ID.PC+4 = 00000004, IF/ID.IR = 00412820

ID/EX registers
ID/EX.rs =  0, ID/EX.rt =  0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs =  0, EX/MEM.rt =  0
EX/MEM.ALUOut = xxxxxxxx, EX/MEM.ALUout = xxxxxxxx

```
        EX/MEM.op = 00

MEM/WB registers
        MEM/WB.rd =  0, MEM/WB.rt =  0
        MEM/WB.value = xxxxxxxx
        EX/MEM.op = 00



clock cycle =  2 (time = 20)
IF/ID registers
        IF/ID.PC+4 = 00000008, IF/ID.IR = 8ca30004

ID/EX registers
        ID/EX.rs =  2, ID/EX.rt =  1
        ID/EX.A = 00000002, ID/EX.B = 00000001
        ID/EX.op = 00

EX/MEM registers
        EX/MEM.rs =  2, EX/MEM.rt =  1
        EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
        EX/MEM.op = 00

MEM/WB registers
        MEM/WB.rd =  0, MEM/WB.rt =  0
        MEM/WB.value = xxxxxxxx
        EX/MEM.op = 00



clock cycle =  3 (time = 30)
IF/ID registers
        IF/ID.PC+4 = 0000000c, IF/ID.IR = 8c420000

ID/EX registers
        ID/EX.rs =  5, ID/EX.rt =  3
        ID/EX.A = 00000005, ID/EX.B = 00000003
        ID/EX.op = 23

EX/MEM registers
        EX/MEM.rs =  5, EX/MEM.rt =  3
        EX/MEM.ALUOut = 00000003, EX/MEM.ALUout = 00000001
        EX/MEM.op = 00
```

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 4 (time = 40)
IF/ID registers
IF/ID.PC+4 = 00000010, IF/ID.IR = 00a31825

ID/EX registers
ID/EX.rs = 2, ID/EX.rt = 2
ID/EX.A = 00000002, ID/EX.B = 00000002
ID/EX.op = 23

EX/MEM registers
EX/MEM.rs = 2, EX/MEM.rt = 2
EX/MEM.ALUOut = 00000009, EX/MEM.ALUout = 00000003
EX/MEM.op = 23

MEM/WB registers
MEM/WB.rd = 5, MEM/WB.rt = 1
MEM/WB.value = 00000003
EX/MEM.op = 00

clock cycle = 5 (time = 50)
IF/ID registers
IF/ID.PC+4 = 00000014, IF/ID.IR = aca30000

ID/EX registers
ID/EX.rs = 5, ID/EX.rt = 3
ID/EX.A = 00000005, ID/EX.B = 00000003
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 5, EX/MEM.rt = 3
EX/MEM.ALUOut = 00000002, EX/MEM.ALUout = 00000002
EX/MEM.op = 23

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 3
MEM/WB.value = fffffff0
EX/MEM.op = 23

clock cycle = 6 (time = 60)
IF/ID registers
IF/ID.PC+4 = 00000018, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 5, ID/EX.rt = 3
ID/EX.A = 00000003, ID/EX.B = 00000003
ID/EX.op = 2b

EX/MEM registers
EX/MEM.rs = 5, EX/MEM.rt = 3
EX/MEM.ALUOut = 00000007, EX/MEM.ALUout = 00000003
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 2
MEM/WB.value = 00000000
EX/MEM.op = 23

clock cycle = 7 (time = 70)
IF/ID registers
IF/ID.PC+4 = 0000001c, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000003, EX/MEM.ALUout = 00000003
EX/MEM.op = 2b

MEM/WB registers

MEM/WB.rd =  3, MEM/WB.rt =  3
MEM/WB.value = 00000007
EX/MEM.op = 00

clock cycle =  8 (time = 80)
IF/ID registers
IF/ID.PC+4 = 00000020, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs =  0, ID/EX.rt =  0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs =  0, EX/MEM.rt =  0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd =  0, MEM/WB.rt =  3
MEM/WB.value = 00000007
EX/MEM.op = 2b

clock cycle =  9 (time = 90)
IF/ID registers
IF/ID.PC+4 = 00000024, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs =  0, ID/EX.rt =  0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs =  0, EX/MEM.rt =  0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd =  0, MEM/WB.rt =  0

MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 10 (time = 100)
IF/ID registers
IF/ID.PC+4 = 00000028, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 11 (time = 110)
IF/ID registers
IF/ID.PC+4 = 0000002c, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000

EX/MEM.op = 00

clock cycle = 12 (time = 120)
IF/ID registers
IF/ID.PC+4 = 00000030, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 13 (time = 130)
IF/ID registers
IF/ID.PC+4 = 00000034, IF/ID.IR = 00000020

ID/EX registers
ID/EX.rs = 0, ID/EX.rt = 0
ID/EX.A = 00000000, ID/EX.B = 00000000
ID/EX.op = 00

EX/MEM registers
EX/MEM.rs = 0, EX/MEM.rt = 0
EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
EX/MEM.op = 00

MEM/WB registers
MEM/WB.rd = 0, MEM/WB.rt = 0
MEM/WB.value = 00000000
EX/MEM.op = 00

clock cycle = 14 (time = 140)
IF/ID registers
	IF/ID.PC+4 = 00000038, IF/ID.IR = 00000020

ID/EX registers
	ID/EX.rs =  0, ID/EX.rt =  0
	ID/EX.A = 00000000, ID/EX.B = 00000000
	ID/EX.op = 00

EX/MEM registers
	EX/MEM.rs =  0, EX/MEM.rt =  0
	EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
	EX/MEM.op = 00

MEM/WB registers
	MEM/WB.rd =  0, MEM/WB.rt =  0
	MEM/WB.value = 00000000
	EX/MEM.op = 00

clock cycle = 15 (time = 150)
IF/ID registers
	IF/ID.PC+4 = 0000003c, IF/ID.IR = 00000020

ID/EX registers
	ID/EX.rs =  0, ID/EX.rt =  0
	ID/EX.A = 00000000, ID/EX.B = 00000000
	ID/EX.op = 00

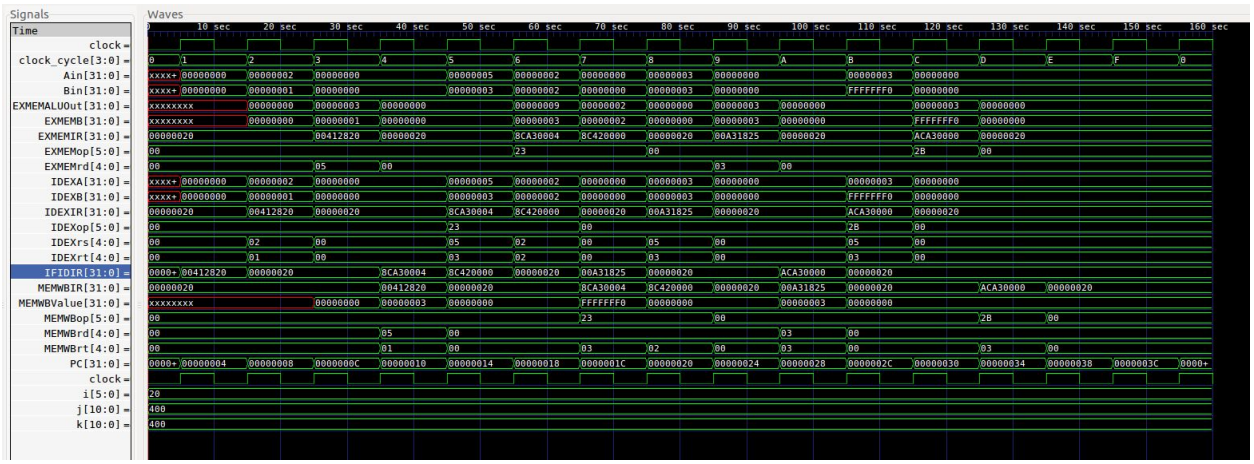
EX/MEM registers
	EX/MEM.rs =  0, EX/MEM.rt =  0
	EX/MEM.ALUOut = 00000000, EX/MEM.ALUout = 00000000
	EX/MEM.op = 00

MEM/WB registers
	MEM/WB.rd =  0, MEM/WB.rt =  0
	MEM/WB.value = 00000000
	EX/MEM.op = 00

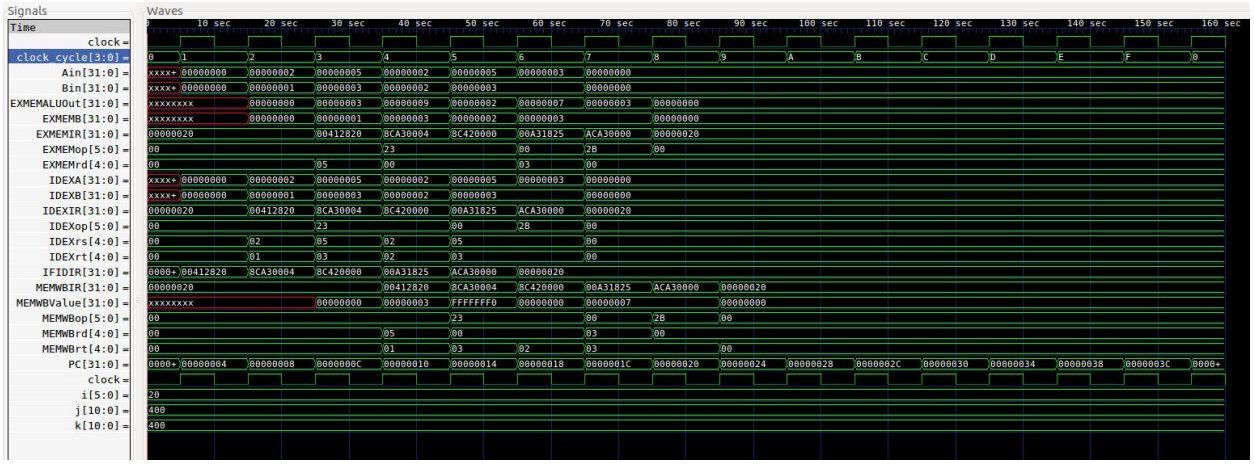
clock cycle =  0 (time = 160)

gtkwave compare:

With Nop:




Without Nop:




Conclusion: The code with bubbles obvious have more cycles than the code without bubbles, that's because the Nop instruction also need 1 cycle to do it, Without Nop code takes 9 cycles but With Nop code takes E(14) cycles, the extra 5 cycles come from the Nop.
The change of Ain with nop: 0-2-5-2-3-3 Ain without nop: 0-2-5-2-5-3 so there between line 3 and 4 have to wait, cause this is the data hazard caused by data dependency. Also in EXMEMALUOut: with nop: 3-9-2-3-3, without nop: 3-9-2-7-3 shows the hazard occurred between line 3 and line 4.
With Nop cycle 12: the Memory execute stage can access memory address FFFFFFF0, but in the witout nop code, it can't access memory FFFFFFF0 in the requirement.

Without Nop Code:

```
// Incomplete behavioral model of MIPS pipeline

module mipspipe(clock);
  // in_out
  input clock;

  // Instruction opcodes
  parameter LW = 6'b100011, SW = 6'b101011, BEQ = 6'b000100, nop = 32'b00000_100000,
ALUop = 6'b0, ADD = 6'b100000, OR = 6'b100101; // Code Modified I
  reg [31:0] PC, // Program counter
                  Regs[0:31],  // Register file
                        IMemory[0:1023], DMemory[0:1023], // Instruction and data memories
          IFIDIR, IDEXA, IDEXB, IDEXIR, EXMEMIR, EXMEMB, // pipeline latches
          EXMEMALUOut, MEMWBValue, MEMWBIR; // pipeline latches

  wire [4:0] IDEXrs, IDEXrt, EXMEMrd, MEMWBrd, MEMWBrt; // fields of pipeline latches
  wire [5:0] EXMEMop, MEMWBop, IDEXop; // opcodes
  wire [31:0] Ain, Bin; // ALU inputs

  // Define fields of pipeline latches
  assign IDEXrs = IDEXIR[25:21]; // rs field
  assign IDEXrt = IDEXIR[20:16]; // rt field
  assign EXMEMrd = EXMEMIR[15:11]; // rd field
  assign MEMWBrd = MEMWBIR[15:11]; // rd field
  assign MEMWBrt = MEMWBIR[20:16]; // rt field -- for loads
  assign EXMEMop = EXMEMIR[31:26]; // opcode
  assign MEMWBop = MEMWBIR[31:26]; // opcode
  assign IDEXop = IDEXIR[31:26]; // opcode

  // Inputs to the ALU come directly from the ID/EX pipeline latches
  assign Ain = IDEXA;
  assign Bin = IDEXB;
  reg [5:0] i; //used to initialize registers
  reg [10:0] j,k; //used to initialize registers

  initial begin
    PC = 0;
    IFIDIR = nop;
    IDEXIR = nop;
    EXMEMIR = nop;
    MEMWBIR = nop; // no-ops placed in pipeline latches
    // test some instructions
        // R-format: op|rs|rt|rd|shmat|func
```

```
        // I-format: op|rs|rt|address
  /*
            add $5,$2,$1 -> 32'b000000|00010|00001|00101|00000|100000
            IMemory[0] = 32'h00412820

            lw $3,4($5) -> 32'b100011|00101|00011|0000000000000100
            IMemory[1] = 32'h8ca30004;

            lw $2,0($2) -> 32'b100011|00010|00010|0000000000000000
            IMemory[2] = 32'h8c420000;

            or $3,$5,$3 -> 32'b000000|00101|00011|00011|00000|100101
            IMemory[3] = 32'h00a31825;

            sw $3,0($5) -> 32'b101011|00101|00011|000000000000000
            IMemory[4] = 32'haca30000;
            for (j=5;j<=1023;j=j+1) IMemory[j] = nop;
  */
  for (i=0;i<=31;i=i+1) Regs[i] = i; // initialize registers
    /*
    IMemory[0] = 32'h8c210003;
    IMemory[1] = 32'hac020000;
    IMemory[2] = 32'h00642820;
        */
        // Code Modified
  IMemory[0] = 32'h00412820;//add $5,$2,$1
  IMemory[1] = 32'h8ca30004;//lw $3,4($5)
  IMemory[2] = 32'h8c420000;//lw $2,0($2)
  IMemory[3] = 32'h00a31825;//or $3,$5,$3
  IMemory[4] = 32'haca30000;//sw $3,0($5)
  for (j=5;j<=1023;j=j+1) IMemory[j] = nop;
  //DMemory[0] = 32'h00000000;
  //DMemory[1] = 32'hffffffff;
        // Code Modified
  for (k=0;k<=1023;k=k+1) DMemory[k] = 0;
  DMemory[2] = 32'hfffffff0; //0($2):32'hfffffff0
  DMemory[5] = 32'hffffffff; //4($5):32'hffffffff
end

always @ (posedge clock)
begin
  // FETCH: Fetch instruction & update PC
  IFIDIR <= IMemory[PC>>2];
```

```
        PC <= PC + 4;

    // DECODE: Read registers
    IDEXA <= Regs[IFIDIR[25:21]];
    IDEXB <= Regs[IFIDIR[20:16]]; // get two registers

        IDEXIR <= IFIDIR; // pass along IR

    // EX: Address calculation or ALU operation
    if ((IDEXop==LW) |(IDEXop==SW)) // address calculation
      EXMEMALUOut <= IDEXA +{{16{IDEXIR[15]}}, IDEXIR[15:0]};
    else if (IDEXop==ALUop) begin // ALU operation
      case (IDEXIR[5:0]) // R-type instruction
        32: EXMEMALUOut <= Ain + Bin; // add operation
           37: EXMEMALUOut <= Ain | Bin; // or operation, Code Modified
        default: ; // other R-type operations [to be implemented]
      endcase
    end

    EXMEMIR <= IDEXIR; EXMEMB <= IDEXB; //pass along the IR & B

    // MEM
    if (EXMEMop==ALUop) MEMWBValue <= EXMEMALUOut; //pass along ALU result
    else if (EXMEMop == LW) MEMWBValue <= DMemory[EXMEMALUOut>>2]; // load
    else if (EXMEMop == SW) DMemory[EXMEMALUOut>>2] <=EXMEMB; // store

    MEMWBIR <= EXMEMIR; //pass along IR

    // WB
    if ((MEMWBop==ALUop) & (MEMWBrd != 0)) // update registers if ALU operation and
destination not 0
                Regs[MEMWBrd] <= MEMWBValue; // ALU operation
    else if ((MEMWBop == LW)& (MEMWBrt != 0)) // Update registers if load and destination not
0
                Regs[MEMWBrt] <= MEMWBValue;
  end

endmodule
```

With Nop:

```
// Incomplete behavioral model of MIPS pipeline
```

```
module mipspipe(clock);
  // in_out
  input clock;

  // Instruction opcodes
  parameter LW = 6'b100011, SW = 6'b101011, BEQ = 6'b000100, nop = 32'b00000_100000,
ALUop = 6'b0, ADD = 6'b100000, OR = 6'b100101; // Code Modified I
  reg [31:0] PC, // Program counter
                  Regs[0:31],  // Register file
                        IMemory[0:1023], DMemory[0:1023], // Instruction and data memories
          IFIDIR, IDEXA, IDEXB, IDEXIR, EXMEMIR, EXMEMB, // pipeline latches
          EXMEMALUOut, MEMWBValue, MEMWBIR; // pipeline latches

  wire [4:0] IDEXrs, IDEXrt, EXMEMrd, MEMWBrd, MEMWBrt; // fields of pipeline latches
  wire [5:0] EXMEMop, MEMWBop, IDEXop; // opcodes
  wire [31:0] Ain, Bin; // ALU inputs

  // Define fields of pipeline latches
  assign IDEXrs = IDEXIR[25:21]; // rs field
  assign IDEXrt = IDEXIR[20:16]; // rt field
  assign EXMEMrd = EXMEMIR[15:11]; // rd field
  assign MEMWBrd = MEMWBIR[15:11]; // rd field
  assign MEMWBrt = MEMWBIR[20:16]; // rt field -- for loads
  assign EXMEMop = EXMEMIR[31:26]; // opcode
  assign MEMWBop = MEMWBIR[31:26]; // opcode
  assign IDEXop = IDEXIR[31:26]; // opcode

  // Inputs to the ALU come directly from the ID/EX pipeline latches
  assign Ain = IDEXA;
  assign Bin = IDEXB;
  reg [5:0] i; //used to initialize registers
  reg [10:0] j,k; //used to initialize registers

  initial begin
    PC = 0;
    IFIDIR = nop;
    IDEXIR = nop;
    EXMEMIR = nop;
    MEMWBIR = nop; // no-ops placed in pipeline latches
    // test some instructions
        // R-format: op|rs|rt|rd|shmat|func
        // I-format: op|rs|rt|address
```

```
  /*
          add $5,$2,$1 -> 32'b000000|00010|00001|00101|00000|100000
          IMemory[0] = 32'h00412820

          lw $3,4($5) -> 32'b100011|00101|00011|0000000000000100
          IMemory[1] = 32'h8ca30004;

          lw $2,0($2) -> 32'b100011|00010|00010|0000000000000000
          IMemory[2] = 32'h8c420000;

          or $3,$5,$3 -> 32'b000000|00101|00011|00011|00000|100101
          IMemory[3] = 32'h00a31825;

          sw $3,0($5) -> 32'b101011|00101|00011|000000000000000
          IMemory[4] = 32'haca30000;
          for (j=5;j<=1023;j=j+1) IMemory[j] = nop;
  */
  for (i=0;i<=31;i=i+1) Regs[i] = i; // initialize registers
    /*
       IMemory[0] = 32'h8c210003;
    IMemory[1] = 32'hac020000;
    IMemory[2] = 32'h00642820;
       */
       // Code Modified
  IMemory[0] = 32'h00412820;//add $5,$2,$1
  IMemory[1] = nop;//insert bubble
  IMemory[2] = nop;//insert bubble
  IMemory[3] = 32'h8ca30004;//lw $3,4($5)
  IMemory[4] = 32'h8c420000;//lw $2,0($2)
  IMemory[5] = nop;//lw $2,0($2) has taken 1 lie, so only need 1 bubble here
  IMemory[6] = 32'h00a31825;//or $3,$5,$3
  IMemory[7] = nop;// insert bubble
  IMemory[8] = nop;// insert bubble
  IMemory[9] = 32'haca30000;//sw $3,0($5)
  for (j=10;j<=1023;j=j+1) IMemory[j] = nop;
  //DMemory[0] = 32'h00000000;
  //DMemory[1] = 32'hffffffff;
       // Code Modified
  for (k=2;k<=1023;k=k+1) DMemory[k] = 0;
  DMemory[2] = 32'hfffffff0; //0($2):32'hfffffff0
  DMemory[5] = 32'hffffffff; //4($5):32'hffffffff
end
```

```
  always @ (posedge clock)
  begin
    // FETCH: Fetch instruction & update PC
    IFIDIR <= IMemory[PC>>2];
          PC <= PC + 4;

    // DECODE: Read registers
    IDEXA <= Regs[IFIDIR[25:21]];
    IDEXB <= Regs[IFIDIR[20:16]]; // get two registers

          IDEXIR <= IFIDIR; // pass along IR

    // EX: Address calculation or ALU operation
    if ((IDEXop==LW) |(IDEXop==SW)) // address calculation
      EXMEMALUOut <= IDEXA +{{16{IDEXIR[15]}}, IDEXIR[15:0]};
    else if (IDEXop==ALUop) begin // ALU operation
      case (IDEXIR[5:0]) // R-type instruction
        32: EXMEMALUOut <= Ain + Bin; // add operation
           37: EXMEMALUOut <= Ain | Bin; // or operation, Code Modified
        default: ; // other R-type operations [to be implemented]
      endcase
    end

    EXMEMIR <= IDEXIR; EXMEMB <= IDEXB; //pass along the IR & B

    // MEM
    if (EXMEMop==ALUop) MEMWBValue <= EXMEMALUOut; //pass along ALU result
    else if (EXMEMop == LW) MEMWBValue <= DMemory[EXMEMALUOut>>2]; // load
    else if (EXMEMop == SW) DMemory[EXMEMALUOut>>2] <=EXMEMB; // store

    MEMWBIR <= EXMEMIR; //pass along IR

    // WB
    if ((MEMWBop==ALUop) & (MEMWBrd != 0)) // update registers if ALU operation and
destination not 0
                Regs[MEMWBrd] <= MEMWBValue; // ALU operation
    else if ((MEMWBop == LW)& (MEMWBrt != 0)) // Update registers if load and destination not
0
                Regs[MEMWBrt] <= MEMWBValue;
   end

endmodule
```