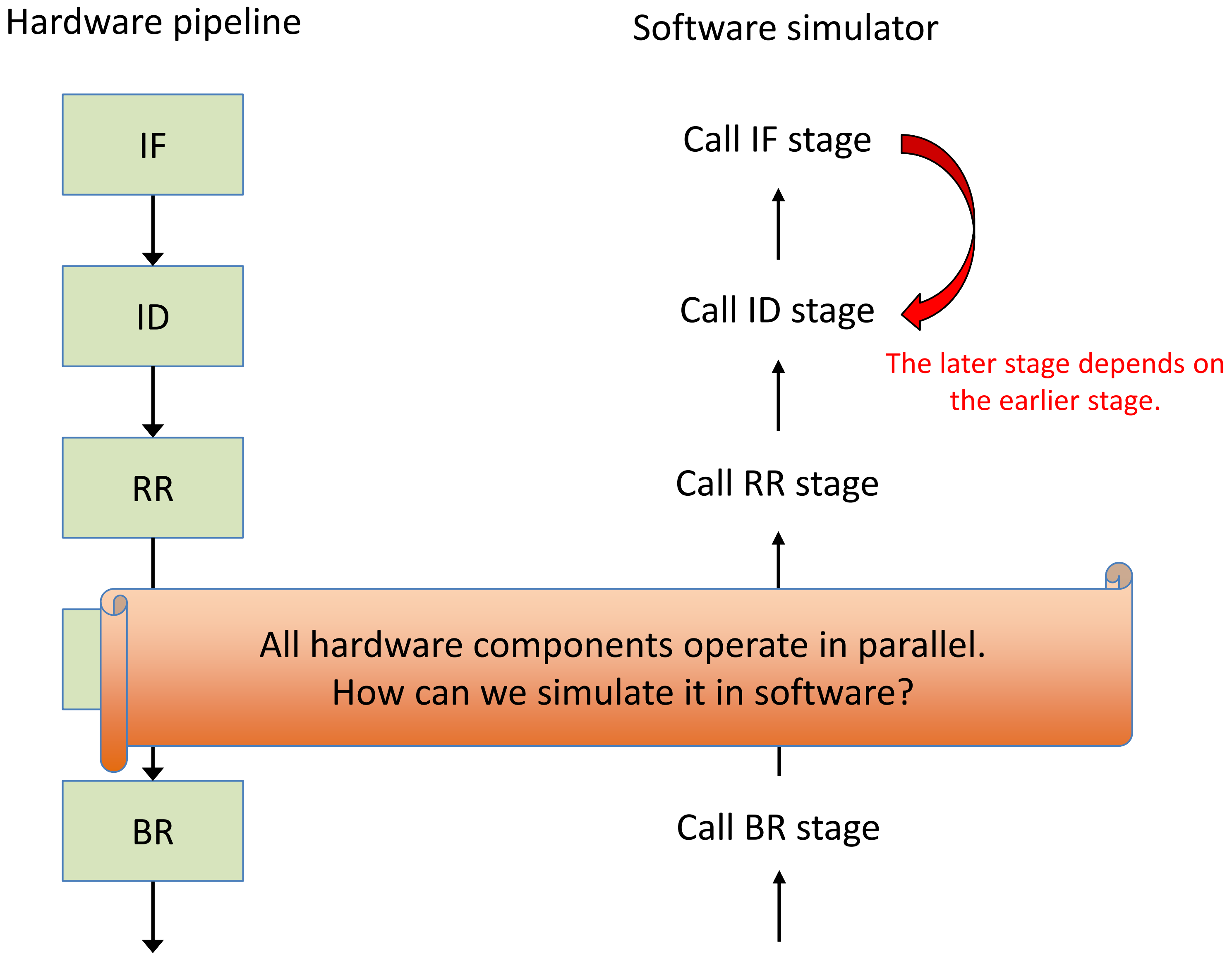
CS 520 Computer Architecture

Project #3

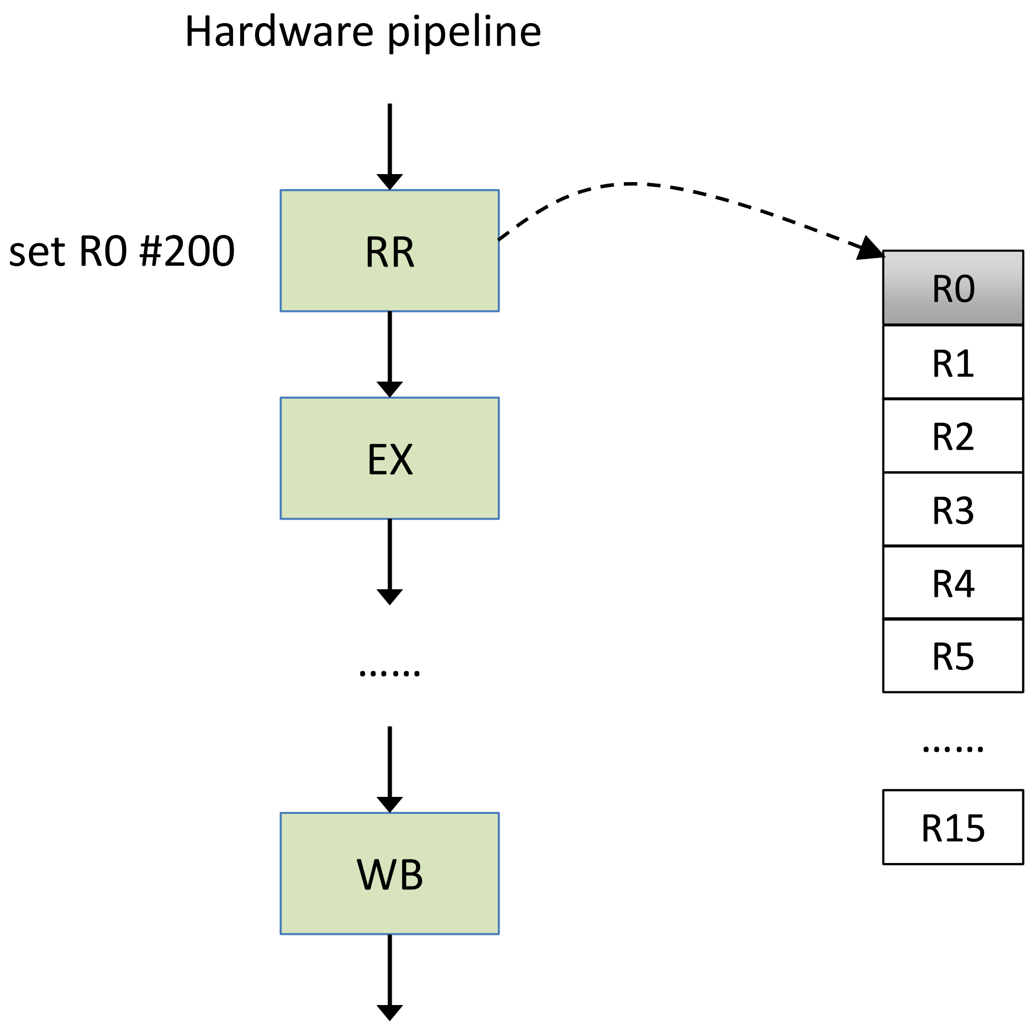
Programming for CPU simulator



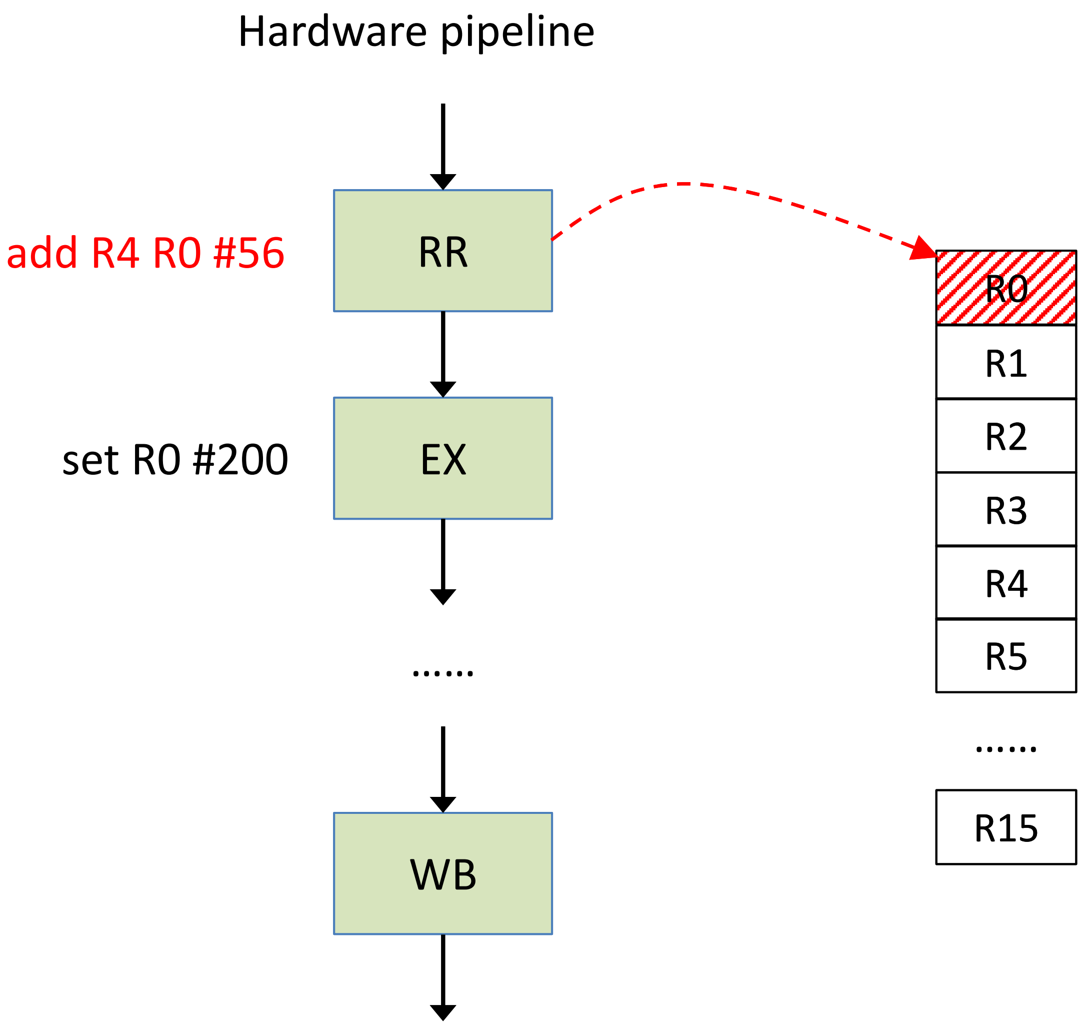
The later stage depends on the earlier stage.

All hardware components operate in parallel.
How can we simulate it in software?

Data Hazard in Pipeline – Cycle 1

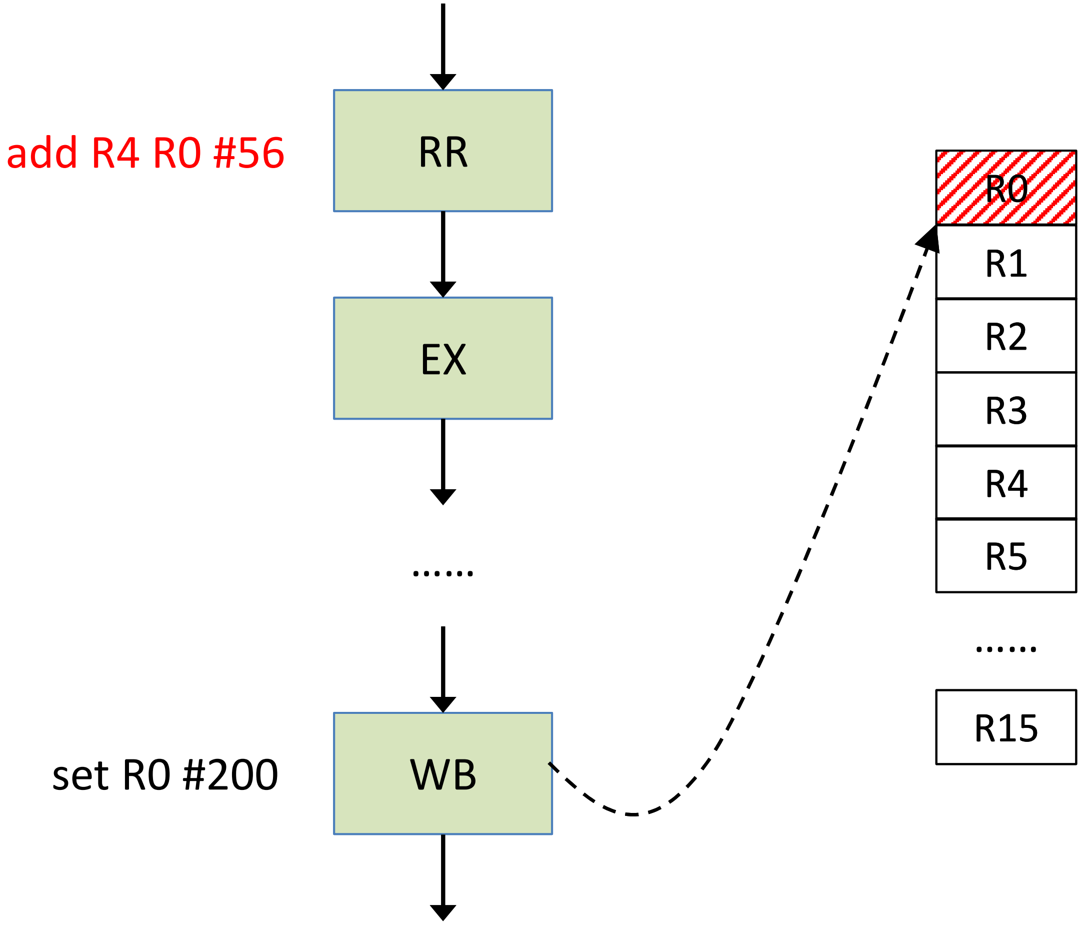


Data Hazard in Pipeline – Cycle 2



Data Hazard in Pipeline – Cycle 3

Hardware pipeline



add R4 R0 #56

RR

EX

……

set R0 #200

WB

R0

R1

R2

R3

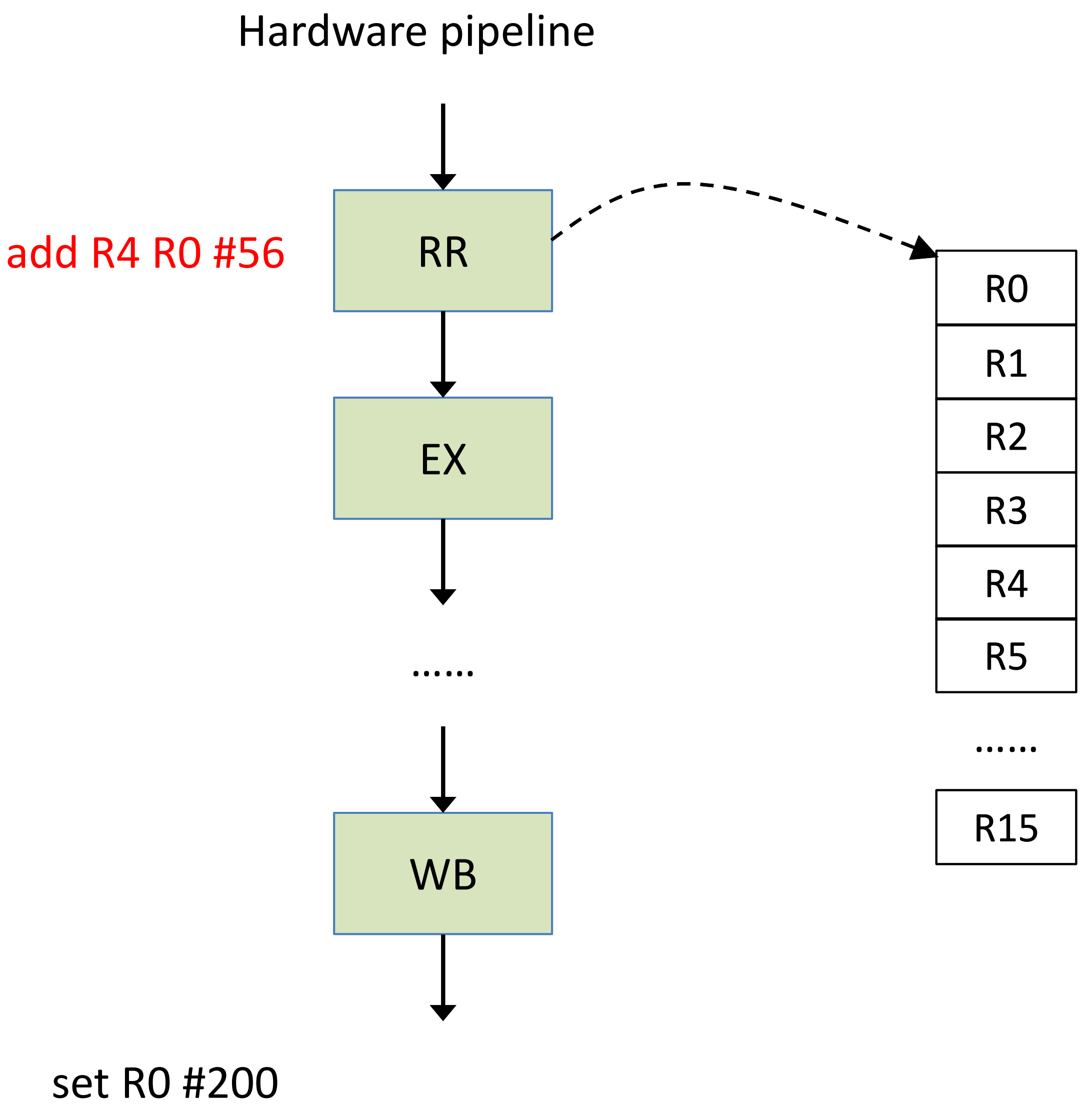
R4

R5

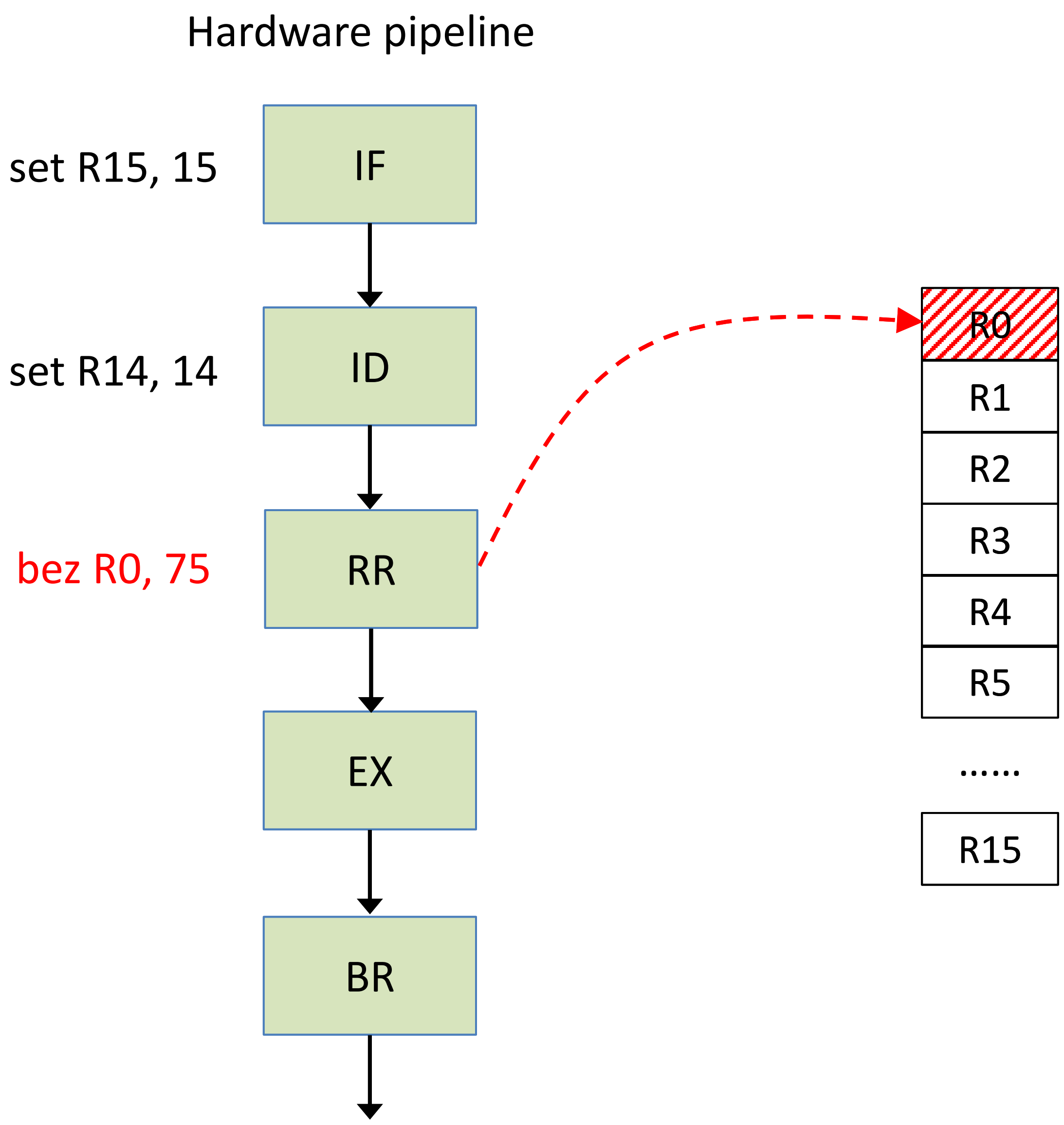
……

R15

Data Hazard in Pipeline – Cycle 4



Control Hazard in Pipeline – Cycle 1



Hardware pipeline

set R15, 15

set R14, 14

bez R0, 75

IF

ID

RR

EX

BR

R0

R1

R2

R3

R4

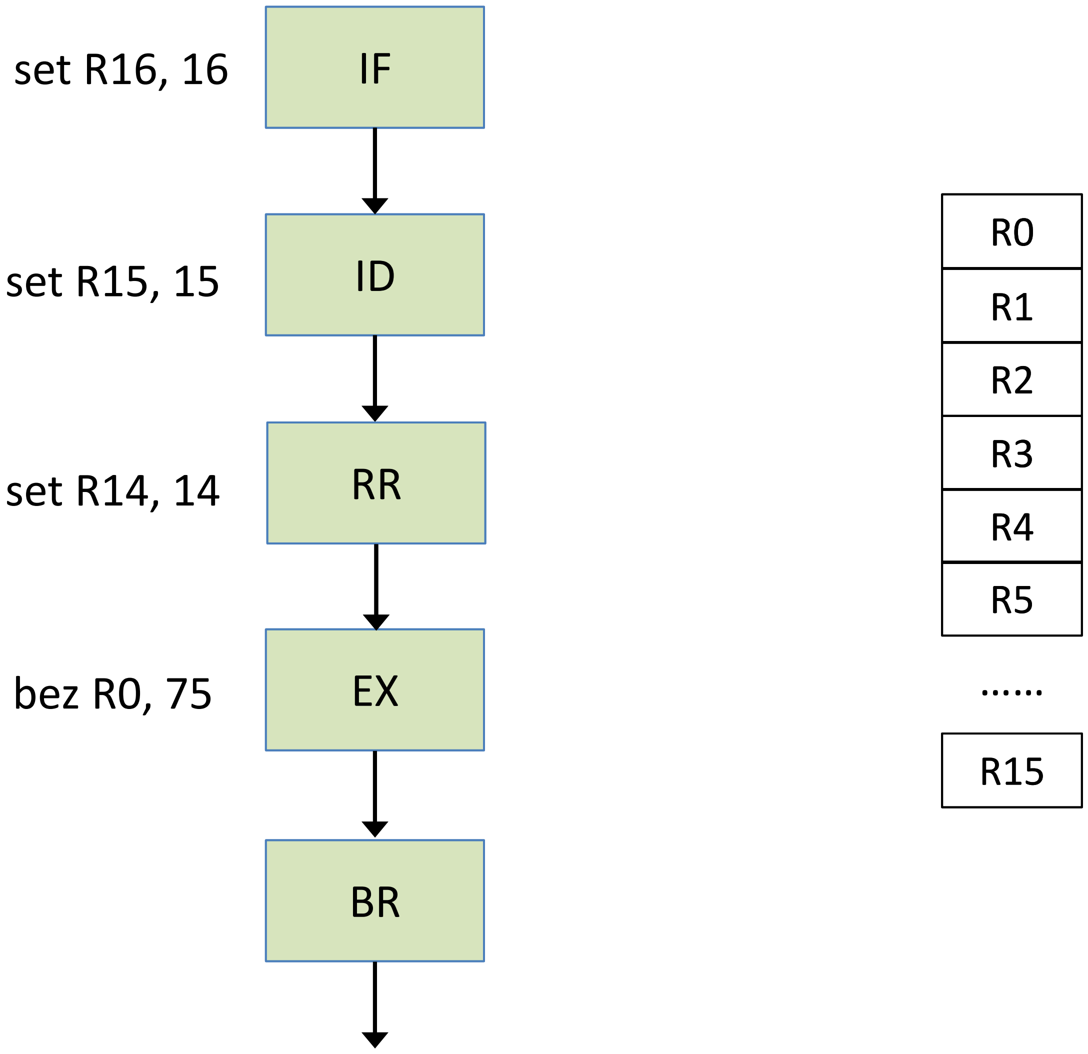
R5

……

R15

Control Hazard in Pipeline – Cycle 2

Hardware pipeline



set R16, 16

IF

set R15, 15

ID

set R14, 14

RR

bez R0, 75

EX

BR

R0

R1

R2

R3

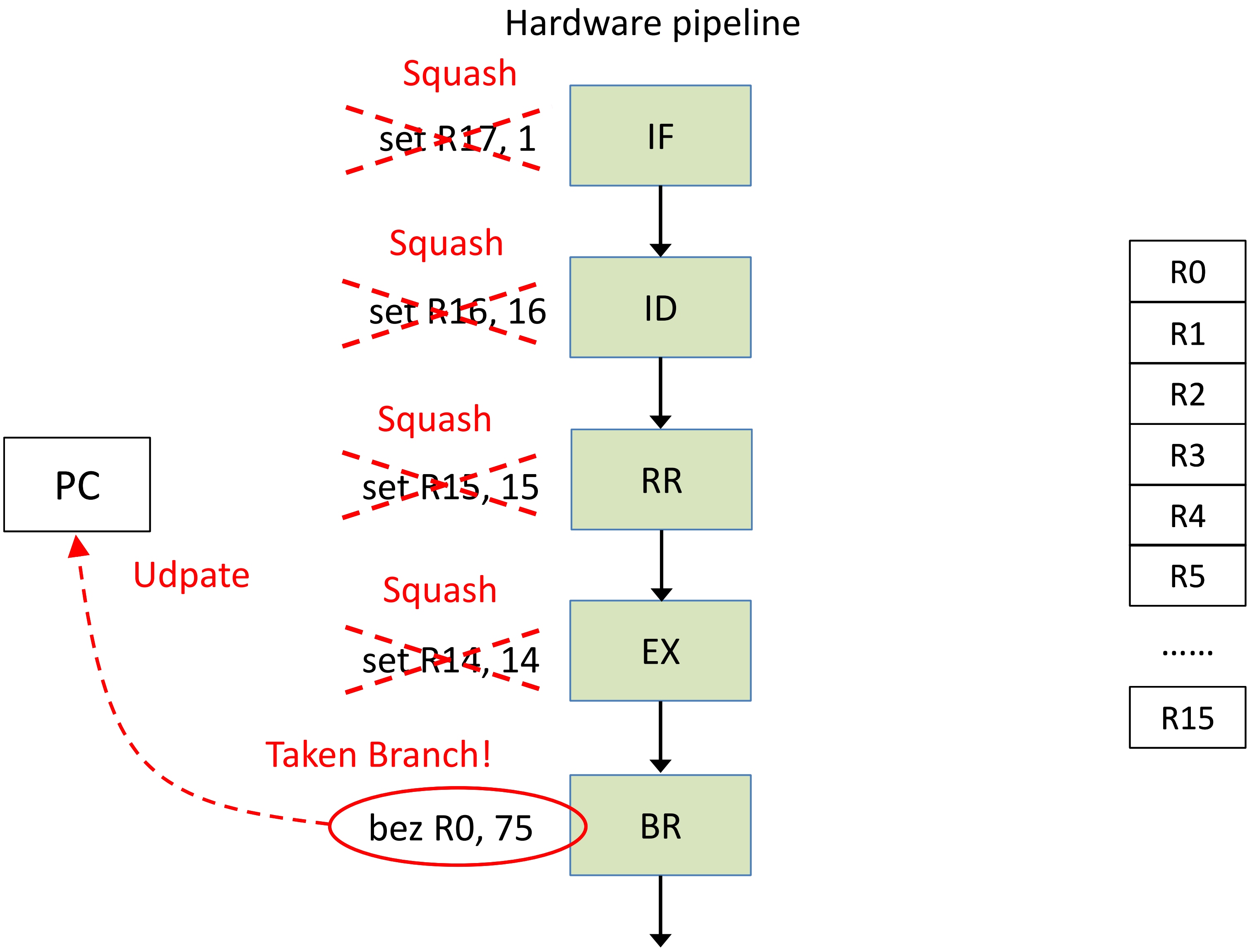
R4

R5

……

R15

Control Hazard in Pipeline – Cycle 3



Hardware pipeline

Squash

~~set R17, 1~~

IF

Squash

~~set R16, 16~~

ID

Squash

~~set R15, 15~~

RR

Squash

~~set R14, 14~~

EX

Taken Branch!

bez R0, 75

BR

PC

Udpate

R0

R1

R2

R3

R4

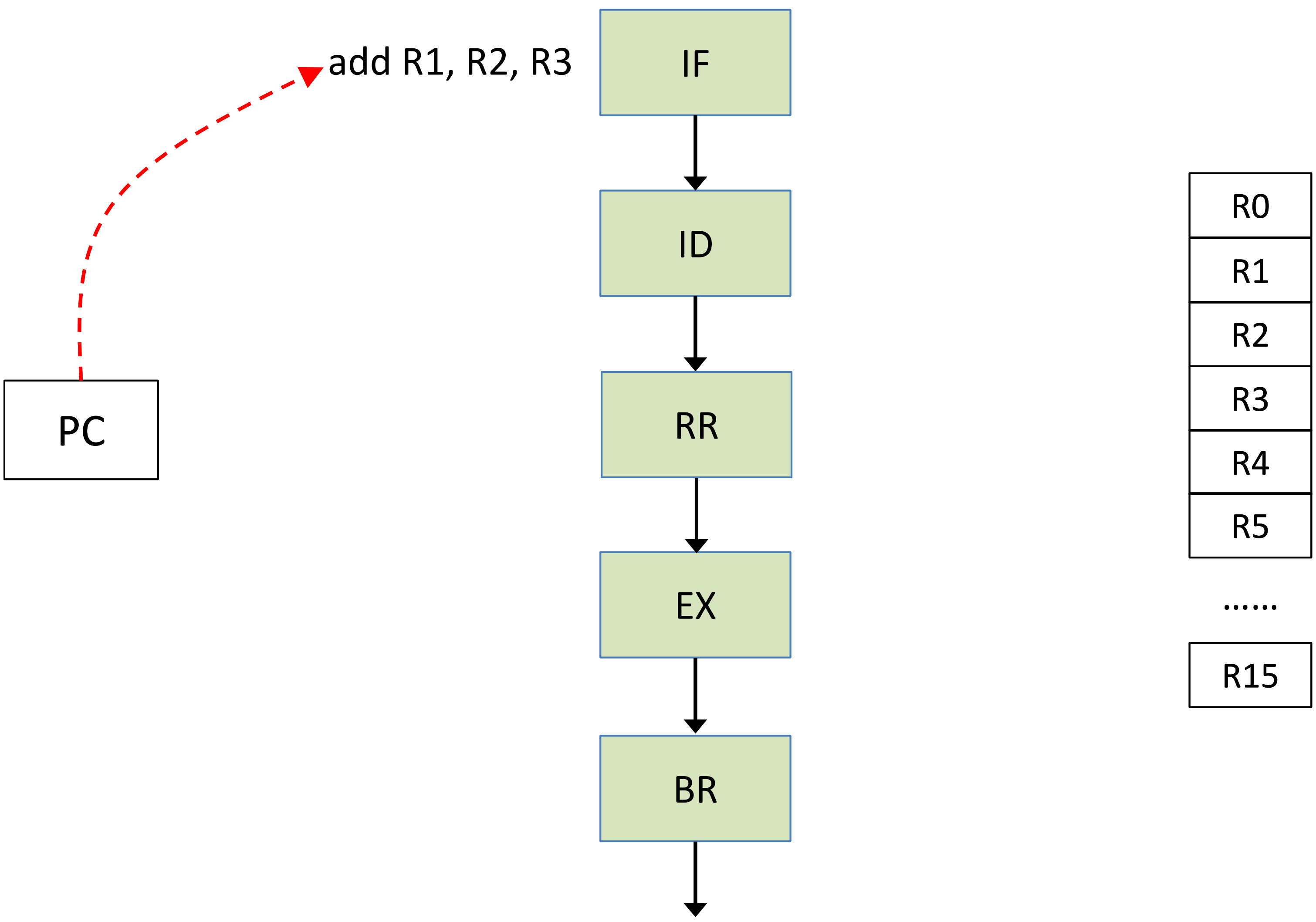
R5

……

R15

Control Hazard in Pipeline – Cycle 4

Hardware pipeline



add R1, R2, R3

PC

IF

ID

RR

EX

BR

R0

R1

R2

R3

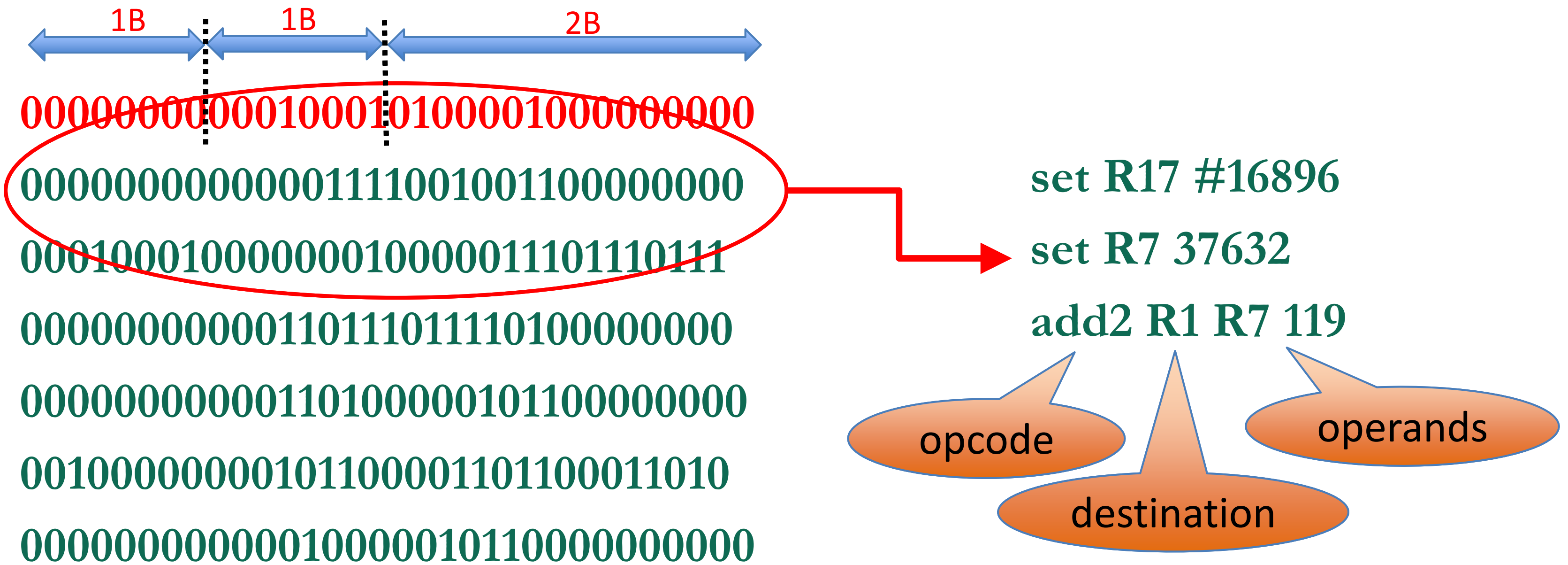
R4

R5

……

R15

Instructions



```
00000000000100010100001000000000
00000000000001111001001100000000
00010001000000010000011101110111
00000000000110111011110100000000
00000000000110100000101100000000
00100000000101100001101100011010
00000000000010000010110000000000
```

set R17 #16896

set R7 37632

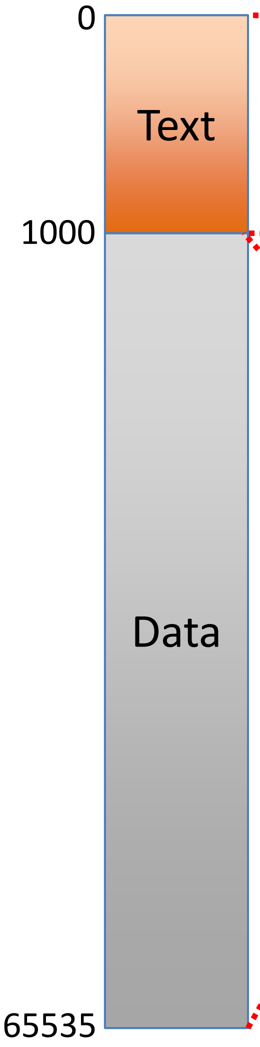
add2 R1 R7 119

....

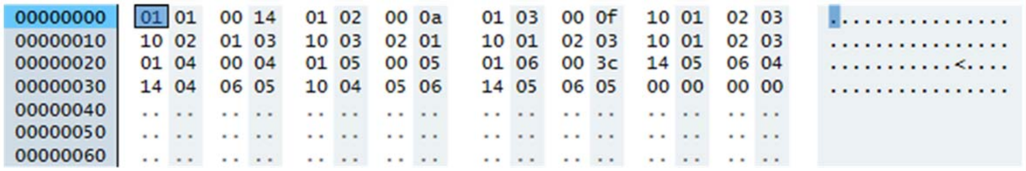
Type	Opcode	Operand 1	Operand 2	Operand 3
Inst-3B	Opcode (1 byte)	Register (1 byte)	Register (1 byte)	-
Inst-4B	Opcode (1 byte)	Register (1 byte)	Register (1 byte)	Register (1 byte)
	Opcode (1 byte)	Register (1 byte)	Immediate (2 bytes)	
Inst-5B	Opcode (1 byte)	Register (1 byte)	Register (1 byte)	Immediate (2 bytes)
Inst-6B	Opcode (1 byte)	Register (1 byte)	Immediate (2 bytes)	Immediate (2 bytes)

- Instructions have varied lengths
- Instructions are in a binary code format
- Instructions located in memory address range from 0 to 3999

Memory Map – mmap#.in

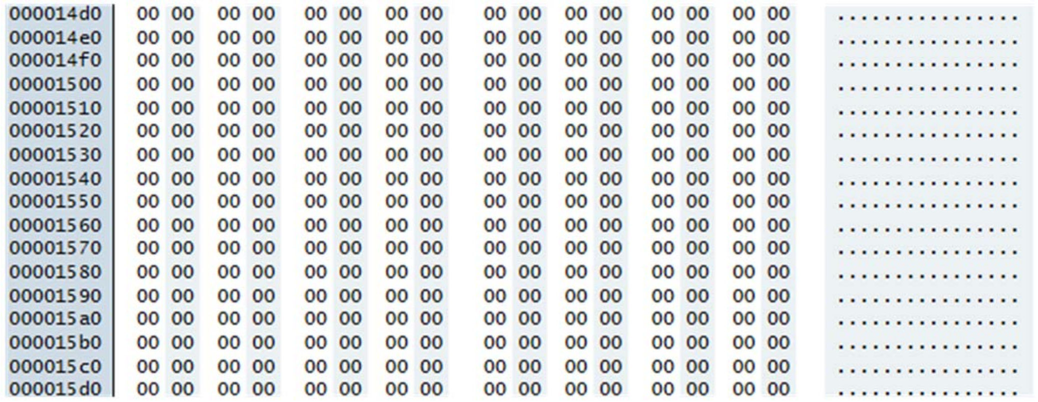


The first 1000B is reserved for programs



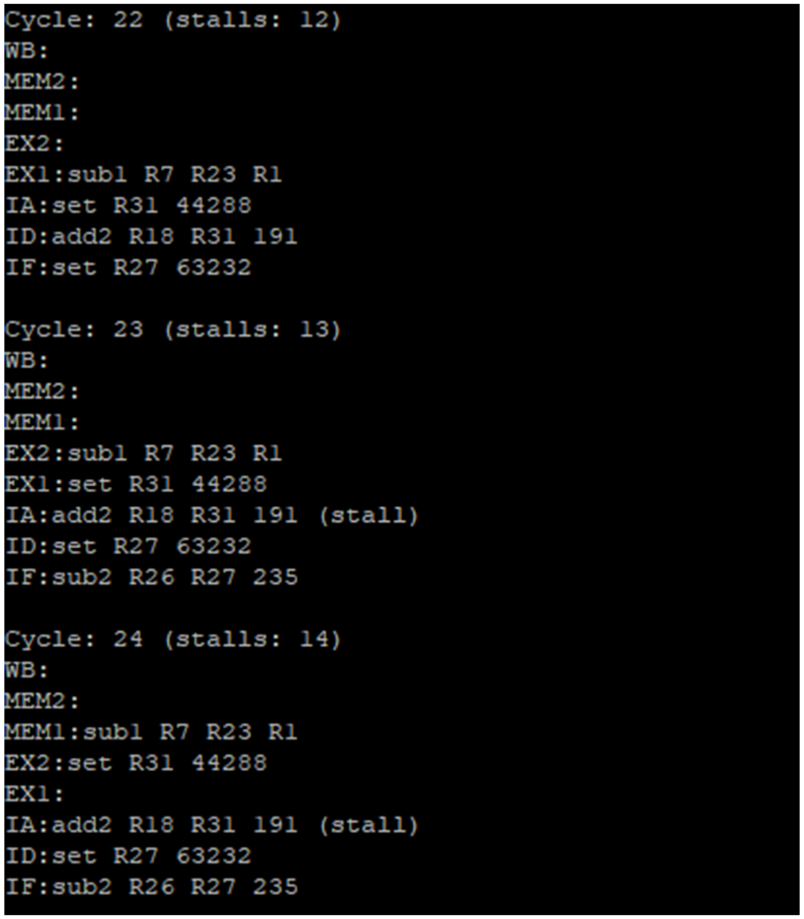
```
00000000  01 01  00 14  01 02  00 0a   01 03  00 0f  10 01  02 03   ................
00000010  10 02  01 03  10 03  02 01   10 01  02 03  10 01  02 03   ................
00000020  01 04  00 04  01 05  00 05   01 06  00 3c  14 05  06 04   ...........<....
00000030  14 04  06 05  10 04  05 06   14 05  06 05  00 00  00 00   ................
00000040  .. ..  .. ..  .. ..  .. ..   .. ..  .. ..  .. ..  .. ..
00000050  .. ..  .. ..  .. ..  .. ..   .. ..  .. ..  .. ..  .. ..
00000060  .. ..  .. ..  .. ..  .. ..   .. ..  .. ..  .. ..  .. ..
```

The memory from 4000 to 65535 is reserved for data.



```
000014d0  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
000014e0  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
000014f0  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001500  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001510  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001520  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001530  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001540  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001550  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001560  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001570  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001580  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
00001590  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
000015a0  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
000015b0  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
000015c0  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
000015d0  00 00  00 00  00 00  00 00   00 00  00 00  00 00  00 00   ................
```

Logs – test#.log



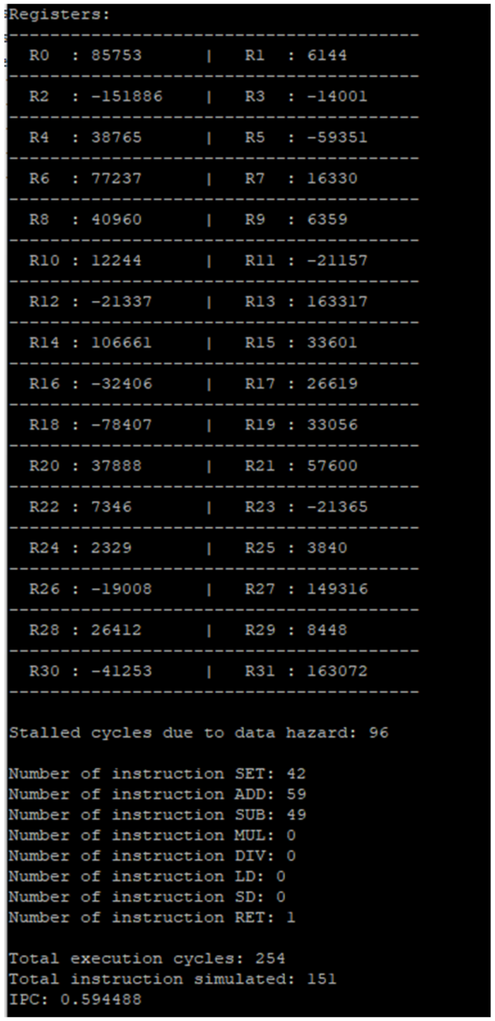
```
Cycle: 22 (stalls: 12)
WB:
MEM2:
MEM1:
EX2:
EX1:sub1 R7 R23 R1
IA:set R31 44288
ID:add2 R18 R31 191
IF:set R27 63232

Cycle: 23 (stalls: 13)
WB:
MEM2:
MEM1:
EX2:sub1 R7 R23 R1
EX1:set R31 44288
IA:add2 R18 R31 191 (stall)
ID:set R27 63232
IF:sub2 R26 R27 235

Cycle: 24 (stalls: 14)
WB:
MEM2:
MEM1:sub1 R7 R23 R1
EX2:set R31 44288
EX1:
IA:add2 R18 R31 191 (stall)
ID:set R27 63232
IF:sub2 R26 R27 235
```

- test#.log shows the execution process during the simulation.
- This file is for debugging.
- Your simulator does not need to produce this file.

Output – output#.txt (sample)



```
Registers:
----------------------------------------
  R0  : 85753       |   R1  : 6144
----------------------------------------
  R2  : -151886     |   R3  : -14001
----------------------------------------
  R4  : 38765       |   R5  : -59351
----------------------------------------
  R6  : 77237       |   R7  : 16330
----------------------------------------
  R8  : 40960       |   R9  : 6359
----------------------------------------
  R10 : 12244       |   R11 : -21157
----------------------------------------
  R12 : -21337      |   R13 : 163317
----------------------------------------
  R14 : 106661      |   R15 : 33601
----------------------------------------
  R16 : -32406      |   R17 : 26619
----------------------------------------
  R18 : -78407      |   R19 : 33056
----------------------------------------
  R20 : 37888       |   R21 : 57600
----------------------------------------
  R22 : 7346        |   R23 : -21365
----------------------------------------
  R24 : 2329        |   R25 : 3840
----------------------------------------
  R26 : -19008      |   R27 : 149316
----------------------------------------
  R28 : 26412       |   R29 : 8448
----------------------------------------
  R30 : -41253      |   R31 : 163072
----------------------------------------

Stalled cycles due to data hazard: 96

Number of instruction SET: 42
Number of instruction ADD: 59
Number of instruction SUB: 49
Number of instruction MUL: 0
Number of instruction DIV: 0
Number of instruction LD: 0
Number of instruction SD: 0
Number of instruction RET: 1

Total execution cycles: 254
Total instruction simulated: 151
IPC: 0.594488
```

- output#.txt shows the final results that your simulator produces.
- Your simulator prints these results on a terminal (standard output).