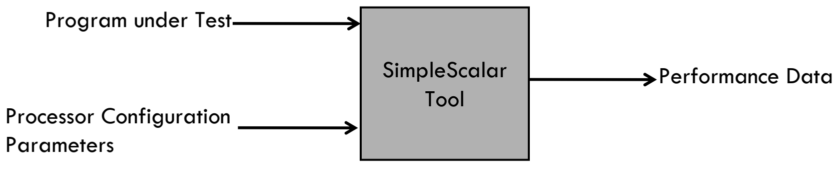
INTRODUCING THE SIMPLESCALAR TOOL SET

Chester Rebeiro
Embedded Lab
IIT Kharagpur

Simplescalar Tool Set

- Modern processors are '*marvels of engineering*' and are increasingly difficult to evaluate
- Simplescalar tool set provides a way to simulate processors built on the Simplescalar architecture



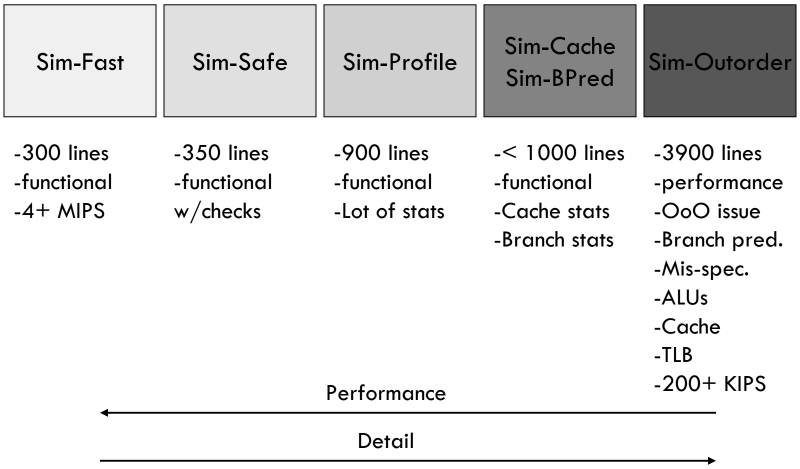
Program under Test

Processor Configuration Parameters

SimpleScalar Tool

Performance Data

Simulator Suite

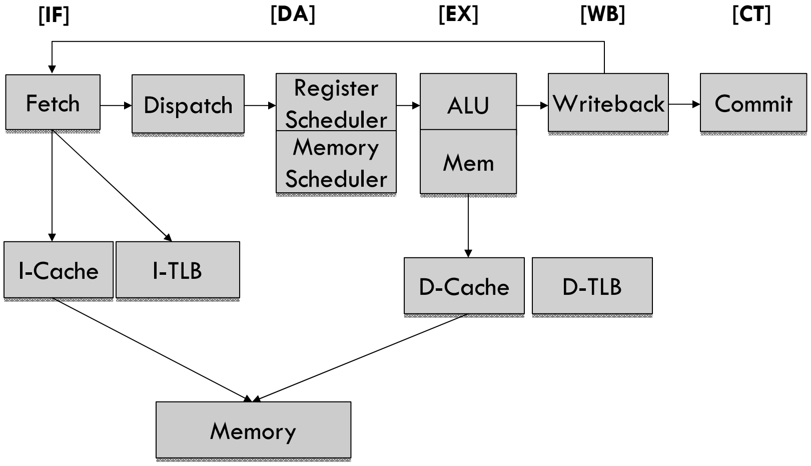


Sim-Fast	Sim-Safe	Sim-Profile	Sim-Cache Sim-BPred	Sim-Outorder
-300 lines -functional -4+ MIPS	-350 lines -functional w/checks	-900 lines -functional -Lot of stats	-< 1000 lines -functional -Cache stats -Branch stats	-3900 lines -performance -OoO issue -Branch pred. -Mis-spec. -ALUs -Cache -TLB -200+ KIPS

Performance

Detail

Simplescalar Outorder Pipeline



[IF] [DA] [EX] [WB] [CT]

Fetch

Dispatch

Register Scheduler

Memory Scheduler

ALU

Mem

Writeback

Commit

I-Cache

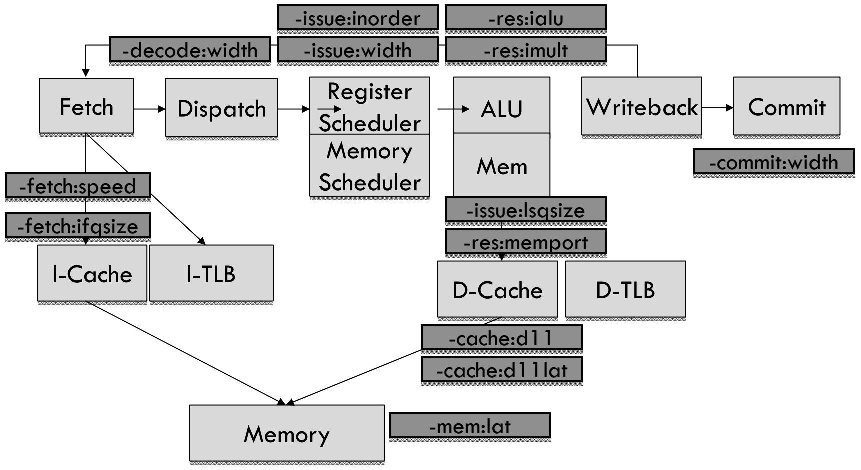
I-TLB

D-Cache

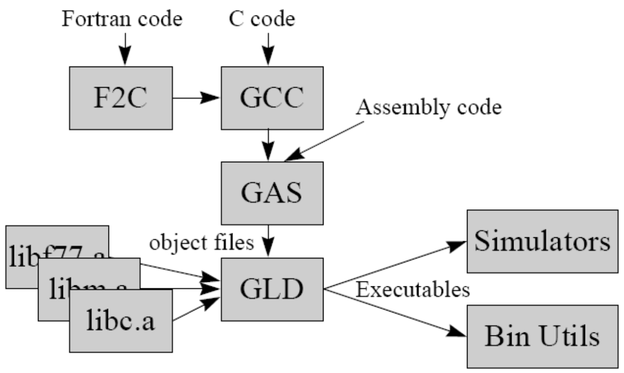
D-TLB

Memory

Configurable Pipeline



Building Targets for the Simplescalar



The ‘Hello World’ Example

- Write the c-program : ‘hello.c’
- Cross-compile it as follows:
  - sslittle-na-sstrix-gcc hello.c
  - This outputs ‘a.out’ compiled for simplescalar
- Obtain the default configuration of the simplescalar
  - ssim-outorder –dumpconfig soo.cfg
  - The default configuration is stored in the file soo.cfg
- Executing a.out
  - ssim-outorder –config soo.cfg ./a.out

Tracing the Pipeline

- produces detailed history of all insts executed, including:
  - instruction fetch, retirement. and pipeline stage transitions
  - supported by sim-outorder
  - enabled via the “-ptrace” option: `-ptrace <file> <range>`
  - useful for pipeline visualization, micro-validation, debugging
- example usage:

```
-ptrace FOO.trc              - trace everything to file FOO.trc
-ptrace BAR.trc 100:5000     - trace from inst 100 to 5000
-ptrace UXXE.trc :10000      - trace until instruction 10000
```

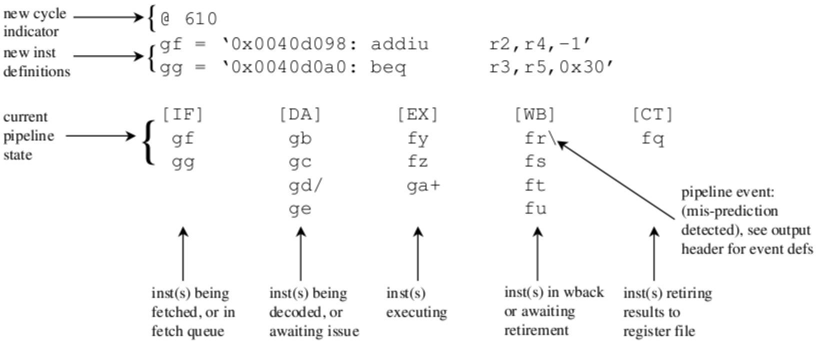
- view with the pipeview.pl Perl script
  - it displays the pipeline for each cycle of execution traced
  - usage: `pipeview.pl <ptrace_file>`

Example ptrace Output

- example session:

```
sim-outorder -ptrace FOO.trc :1000 test-math
pipeview.pl FOO.trc
```

- example output:



```
@ 610
gf = ‘0x0040d098: addiu     r2,r4,-1’
gg = ‘0x0040d0a0: beq       r3,r5,0x30’

 [IF]      [DA]      [EX]      [WB]      [CT]
  gf        gb        fy        fr\       fq
  gg        gc        fz        fs
            gd/       ga+       ft
            ge                  fu
```

new cycle indicator

new inst definitions

current pipeline state

inst(s) being fetched, or in fetch queue

inst(s) being decoded, or awaiting issue

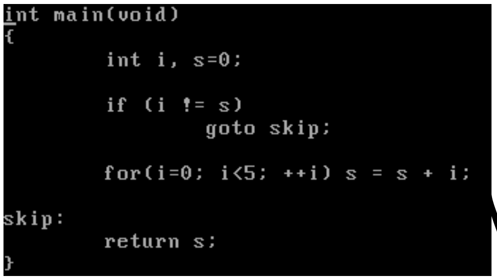
inst(s) executing

inst(s) in wback or awaiting retirement

inst(s) retiring results to register file

pipeline event: (mis-prediction detected), see output header for event defs

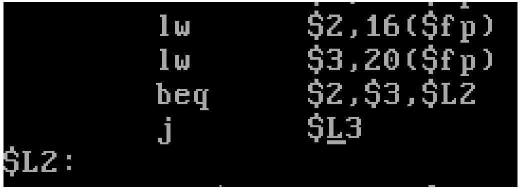
Viewing Control Hazards



```
int main(void)
{
        int i, s=0;

        if (i != s)
                goto skip;

        for(i=0; i<5; ++i) s = s + i;

skip:
        return s;
}
```

Generate assembly code by compiling as follows

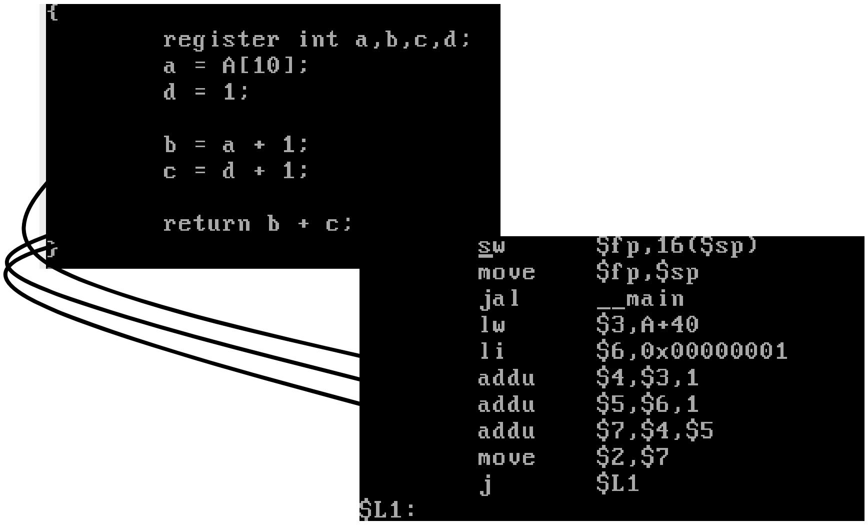
$ sslittle-na-sstrix-gcc control.c –S
And
$sslittle-na-sstrix-objdump --disassemble control.c



```
        lw      $2,16($fp)
        lw      $3,20($fp)
        beq     $2,$3,$L2
        j       $L3
$L2:
```

Use ‘not taken’ branch prediction

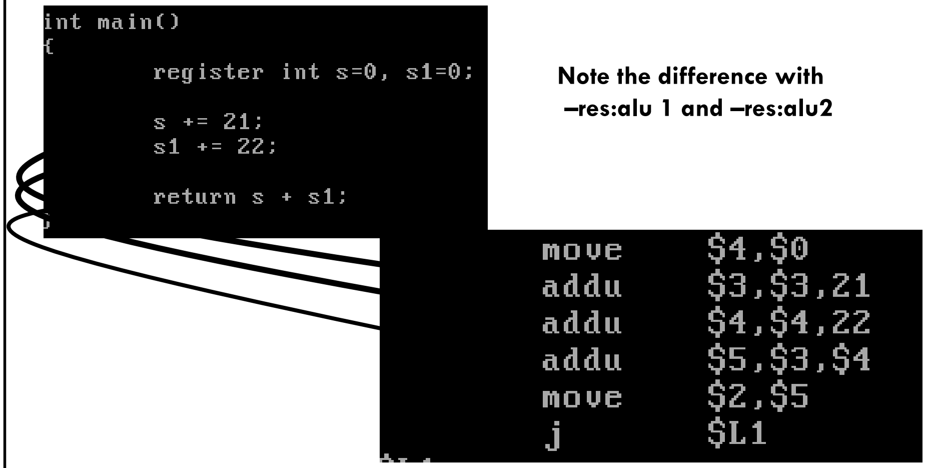
Viewing Data Hazards



```
{
        register int a,b,c,d;
        a = A[10];
        d = 1;

        b = a + 1;
        c = d + 1;

        return b + c;
}
```

```
        sw      $fp,16($sp)
        move    $fp,$sp
        jal     __main
        lw      $3,A+40
        li      $6,0x00000001
        addu    $4,$3,1
        addu    $5,$6,1
        addu    $7,$4,$5
        move    $2,$7
        j       $L1
$L1:
```

Viewing Structural Hazard



```
int main()
{
        register int s=0, s1=0;

        s += 21;
        s1 += 22;

        return s + s1;
}
```

```
        move    $4,$0
        addu    $3,$3,21
        addu    $4,$4,22
        addu    $5,$3,$4
        move    $2,$5
        j       $L1
```

Note the difference with –res:alu 1 and –res:alu2

Thank You