332 Advanced Computer Architecture

Exercise 5 (ASSESSED): instruction-level-parallelism in Jacobi2d

This is the first of two equally-weighted assessed coursework exercises. Hand your solution into the Dept of Computing Student General Office (Huxley 345) by 5pm on Tuesday February 18th.

Background

Recall the Jacobi2d kernel:

```
    typedef double gridtype[MAXiDIM][MAXjDIM];
    gridtype grid1;
    gridtype grid2;
    int it,i,j;
    double onequarter = 1.0/4.0;

    InitGrid(grid1, iDim, jDim);
    InitGrid(grid2, iDim, jDim);

    for (it=0; it<Niters; it+=1) {
      for (i=1; i<iDim-1; ++i)
        for (j=1; j<jDim-1; ++j)
          grid2[i][j] = onequarter *
            (grid1[i-1][j] + grid1[i+1][j] + grid1[i][j-1] + grid1[i][j+1]);
      it++;
      if (it >= Niters) break;

      for (i=1; i<iDim-1; ++i)
        for (j=1; j<jDim-1; ++j)
          grid1[i][j] = onequarter *
            (grid2[i-1][j] + grid2[i+1][j] + grid2[i][j-1] + grid2[i][j+1]);
    }
#ifdef PRINT_GRID
    /* the grid to print depends on whether Niters was odd or even */
    PrintGrid(Niters%2==1 ? grid2 : grid1, iDim, jDim);
#endif
```

Running the Jacobi benchmark under the Simplescalar simulator

You can find the source code in

`/homes/phjk/ToyPrograms/C/Jacobi2d-2003/Jacobi2d.c`

Copy this file to your own directory. You should compile the application using the the following command line:

```
~phjk/simplescalar/bin/gcc -O3 -DMAXiDIM=$x -DMAXjDIM=$x -o Jacobi2d Jacobi2d.c
```

where `$x` is a shell variable set to the array size to be used. Good values lie in the range 200-1000. Now run the application as follows:

```
~phjk/simplescalar/simplesim-2.0/sim-outorder ./Jacobi2d $x $x 2
```

Here the value 2 is the number of iterations of the Jacobi sweep to be repeated. Choose this value to keep the simulation time under control while ensuring the simulated machine does a significant amount of work - 2 is the minimum sensible value.

What to do

Your job is to study the effect of various architectural features on the performance of the Jacobi2d application running under the SimpleScalar simulator:

1. Use sim-outorder's "-ruu:size" flag to vary the RUU size between 2 and 256 (only powers of two are valid). Plot a graph showing your results. Explain what you see.
2. Examine the other sim-outorder parameters which determine the CPU's instruction issue rate (leave the cache parameters unchanged). Where is the bottleneck (when running Jacobi2d) in the default simulated architecture? Justify your answer.

Write your results up in a short report (less than four pages including graphs and discussion).

Tools and tips

Running the experiments

To do this you need to write a shell script. You might find the following Bash script useful:

```
#!/bin/sh -f

SIZE=200
ROOT=/homes/phjk/ToyPrograms/C/Jacobi2d-2003

for x in 2 4 8 16 32 64 128 256
do
  echo -n $x "\t"
  ~phjk/simplescalar/simplesim-2.0/sim-outorder -ruu:size $x \
     $ROOT/simplescalar/Jacobi2d-512-O3 $SIZE $SIZE 2 \
     2>&1 | grep sim_IPC
done
```

The output is delivered to the file "results". You can find this script in

```
~phjk/ToyPrograms/C/Jacobi2d-2003/scripts/Jacobi2d-ss-ruu-script.sh
```

See also "Jacobi2d-ss-vary.sh".

Plotting a graph

Try using the gnuplot program. Run the script above, and save the output in a file "table". Type "gnuplot". Then, at its prompt type:

```
set logscale x 2
plot [][0:2] 'table2' using 1:3 with linespoints
```

To save the plot as a postscript file, try:

```
set term postscript eps
set output "psfile.ps"
plot [][0:2] 'table2' using 1:3 with linespoints
```

Try "help postscript", "help plot" etc for further details.

Examining the assembler code

Compile the application using the "-S" flag.

```
~phjk/simplescalar/bin/gcc -O3 -S -DMAXiDIM=$x -DMAXjDIM=$x Jacobi2d.c
```

The assembler code generated by the compiler is delivered in "Jacobi2d.s".

Experimenting with compiler options

The compiler accepts many parameters to control optimisation. Try "man gcc" to read about them. For example, "-funroll-loops" encourages the compiler to unroll loops:

```
~phjk/simplescalar/bin/gcc -O3 -funroll-loops -DMAXiDIM=$x -DMAXjDIM=$x -o Jacobi2d Jacobi2d
```

Does this improve the performance of the simulated machine? What happens to the IPC?

Paul Kelly, Imperial College London, 2003