Universidade de Aveiro

Mestrado Integrado em Engenharia de Computadores e Telemática

Arquitectura de Computadores Avançada

Lição 6: Code optimization

Academic year 2015/2016 — Nuno Lau, José Luís Azevedo

1. The following program adds two vectors of floats and places the result in a third vector.

```
        .data
        .align 4
_x:     .float 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0
_y:     .float 10.0,20.0, 30.0, 40.0, 50.0, 60.0, 70.0, 80.0
_z:     .space 32

        .align 4
        .text
        .global _main
_main:
        lhi   r3, (_y>>16)
        addui r3, r3, (_y&0xffff)
        lhi   r4, (_x>>16)
        addui r4, r4, (_x&0xffff)
        lhi   r5, (_z>>16)
        addui r5, r5, (_z&0xffff)
        addui r6, r3, 32
L5:
        lf    f4, 0(r3)
        lf    f5, 0(r4)
        addf  f4, f4, f5
        sf    0(r5), f4
        add   r3, r3, 4
        add   r4, r4, 4
        add   r5, r5, 4
        sge   r1, r3, r6
        beqz  r1, L5
        trap  0
```

1.1. Draw the dependency graph between the instructions of the program's main loop.

1.2. Simulate the previous program using **`WinDLX`**. Draw the timing diagram for one loop iteration. Check the causes of the various pipeline stalls. Take note of the number of cycles it takes to execute the program.

1.3. Unroll the main loop of the program so that each iteration of the unrolled loop corresponds to 4 iterations of the original sequence. Use different registers in each instance of the original loop sequence.

Take note of the number of clock cycles it takes to execute the program. Determine the speedup obtained by this version, taking as reference the original program.

1.4. Reorder the instructions of the unrolled loop to minimize the number of stalls. Determine the speedup obtained by this version, taking as reference the original program.

1.5. Would it be possible to use this optimization in loops whose number of iterations is unknown at compile time?

2. Consider the following assembly language program for the DLX processor:

```
        .data
a:      .double  3.14159265358979
x:      .double  1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
        .double  17,18,19,20,21,22,23,24,25,26
xtop:   .double  27
y:      .double  10,20,30,40,50,60,70,80,90,100,110,120,130,140
        .double  150,160,170,180,190,200,210,220,230,240,250,260
ytop:   .double  270
        .text
        ld    f8, a(r0)
        add   r5, r0, a
        add   r2, r0, xtop
        add   r3, r0, ytop
loop:   ld    f10, 0(r2)
        ld    f4, 0(r3)
        multd f2, f10, f8
        addd  f6, f2, f4
        sd    0(r2), f6
        sub   r3, r3, 8
        sub   r2, r2, 8
        sub   r4, r2, r5
        bnez  r4, loop
        trap  0
```

2.1. Analyse the program and identify the implemented functionality. Run the program in the **WinDLX** simulator and take note of the number of clock cycles it takes to execute.

2.2. Draw the dependency graph between the instructions of the program's main loop.

2.3. Reorder the program instructions in order to minimize the number of clock cycles it takes to execute. Use the **WinDLX** simulator to verify the correctness of the optimized code. Determine the speedup obtained by this version, taking as reference the original program.

2.4. Optimize the program execution time by unrolling the main loop so that an iteration of the unrolled loop corresponds to 3 iterations of the original version. Use different registers in each of the 3 copies of the original loop.

Reorder the loop instructions and verify the correctness of the resulting code by using the **WinDLX**. Determine the speedup obtained by this version, taking as reference the original program.

2.5. Consider that your processor contains 3 independent floating point multiplication units and 3 floating point adding units. Change the program obtained in the previous exercise in order to reduce its execution time by taking advantage of this new processor configuration. Run the program in the **WinDLX** simulator and discuss the results.

2.6. Modify the original program using software pipelining techniques in order to reduce its execution time. Run the resulting program in the **WinDLX** and take note of the execution time (clock cycles) Determine the speedup obtained by this version, taking as reference the original program.

Bibliography

[1] "Computer Organization and Design", David Patterson and John Hennessy, 2nd Edition, Morgan Kauffman