Compiling for the Risc System/6000 Branch Unit

Martin Hopkins

Watson Research Center

IBM

/

Classical 801 Risc, 1975 - 85

Time = path length * cycles per instruction * cycle time

801 emphasized last two.

- Hard wire the CPU. Avoid microcode.
- Shorten the cycle time by:
  - Regular instruction formats.
  - Simple instructions. (Not fewer)
- Promote operand reuse by:
  - Many general purpose registers.
  - Ops that preserve all operands.
- Increase micro parallelism. (Pipelining)
  - Loads proceed in parallel.
  - Special attention to branches.
- Make subroutine linkage fast.
- Invest in the memory subsystem.
- Rely on optimizing compilers to exploit hardware.
- Compiler writers help specify architecture.

RS/6000 Hardware

The evolution to Post Risc.

- CPU is three CMOS II chips, 32 - 40 ns cycle versions.
  - Branch Unit and Instruction Cache.
  - Fixed Point Unit.
  - Floating Point Unit.
- 64 KB Data cache
  - 128 byte lines
  - Four way set associative
- Data TLB on Fixed point chip.
  - 128 entries
  - Two way set associative
  - Hardware reload.
- 8 KB instruction cache on branch chip.
  - Two way set associative.
- Instruction TLB on branch chip.
  - Two way set associative.
- Main memory, minimum of 8 MB, maximum 256 MB.
- Inverted page tables, 4k pages.

Fixed Point Unit.

- 32 GPRs, 32 bits wide.
- No arithmetic from storage. (Must use loads.)
- (Base + Displacement) and (Base + Index) addressing
- Three output, two input Register File
- Ops are all 32 bits long.
  - All operations are non destructive. RT = RA op RB
  - Full set of immediate operations. RT = RA op $\pm 2^{15}$
- High function ops:
  - Update forms of all loads and stores.
  - String assist ops.
  - Load and store multiple.
  - Single cycle ops to rotate, mask, and extract or insert.
  - 3 - 5 cycle integer multiply.
  - 19 cycle integer divide.
  - DOZ for computing min and max.
  - Traps for HLL checking.

Floating Point Unit

- IEEE floating point format.
- Biased toward double precision.
- Highly pipelined.
- Multiply-Add is basic floating point operation.

  RT = RA x RB + RC

  One such operation can be completed every cycle.
- 32 architected 64 bit FPRs.
- Actually has six more.

  Register renames permits fetching data to a register without overlaying data that is still required.

Branch Unit.

- Prefetches instructions.
- Dispatches two operations per cycle to fixed and float units.

  Any combination.
- Performs all branch processing.

  Fixed and float units do not see branches.
- Eight four-bit condition registers.
  - Set by compare ops in other units.
  - Fixed point ops optionally set CR0.
  - Full set of boolean ops on condition register bits.
- Most branches are relative.
- Link register used for register branches.

  Set by BAL or a copy from the fixed point unit.
- Count register used by BCT op.
  - Subtracts one from count register.
  - Branches on not zero take zero time.
  - Permits counting of loop iterations to be done in the branch unit in parallel with the fixed point unit.

An Example of the use of BCT

```
     do 10 i = 0, 511
       do 10 j = 0, 511
         c(i, j) = 0
         do 10 k = 0, 511
10          c(i, j) = c(i, j) + a(i, k) * b(k, j)
```

Compiled Code for two inner loops.

```
MIDDLE_LOOP:
   LRFL     fp1=fp0
   LR       r12=r10
   LR       r11=r8
   LCTR     R0

LOOP:
   LFDU     fpr3,r11=a(r11,4096)    1
   LFDU     fpr2,r10=b(r10,8)       1
   FMA      fpr1=fpr1,fpr3,fpr2     0   overlapped
   BCT      LOOP                    0   overlapped

   C        cr1=r8,r7
   STFDU    r9,c(r9,4096)
   AI       r8=r8,4096
   BH       MIDDLE_LOOP
```

7

An Example of the use of BCT

```
total cycles for inner loop    2
```

5

```
        C      CRA=
        BE     LABEL1,CRA
        OP1
        B      LABEL2     two cycle delay, unresolved BE
LABEL1:
        OP2
LABEL2:
        OP3
        OP4
        OP5
```

The scheduling phase of the compiler converts the above to:

```
        C      CRA=
        BE     LABEL1,CRA
        OP1
        OP3
        OP4
        B      LABEL2           no delay as BE resolved
LABEL1:
        OP2
        OP3
        OP4
LABEL2:
        OP5
```

An Example from Spec

Inner loop of xlygetvalue in Spec bench mark li.
A pointer chain search problem,

//

An Example from Spec

```
     Original                          After Replication

Loop cycles                         Loop cycles

1 L1: L      R4=0(R5,4)               L1: L      R4=0(R5,4)
1     <load delay>                        <load delay>
1     L      R0=0(R4,4)                   L      R0=0(R4,4)
1     <load delay>                        <load delay>
1     C      CR1=R3,R0                    C      CR1=R3,R0
3     BNE    L2,CR1                 0 L3: BE     L5,CR1
      L      R3=0(R4,8)                      loop cleansed
      BLR                                    of exit code
1 L2: L      R5=0(R5,8)             1     L      R5=0(R5,8)
1     <load delay>                  1     <load delay>
1     C      CR0=R5,0               1     C      CR0=R5,0
3     BNZ    L1,CR0                 0     BZ     L4,CR0
                                    1     L      R4=0(R5,4)
                                    1     <load delay>
                                    1     L      R0=0(R4,4)
                                    1     <load delay>
                                    1     C      CR1=R3,0
                                    0     B      L3
                                      L4:

14 cycles for loop                  8 cycles for loop
```

Branch Swapping

```
loop:
   :
   c    cr100=...
   be   exit,cr100
   bct  loop
```

- The problem is that the BE op can execute in zero cycles, but the BCT must wait until the BE is resolved.
- This is solved by branch swapping.

```
   b    enter
loop:
   be   exit
enter:
   :
   c    cr100=...
   bct  loop
   be   exit,cr100
```

- The two branches in the inner loop now take zero cycles.

Branch Performance

- Branches execute in zero cycles if:
  - They are unconditional.
    In the case of branch on the link register, four cycles must intervene between the setting of the link register from a GPR and the branch.
  - Conditional branches that fall through.
  - When taken, there are three ops between the compare and the branch.
    Six for float compare.
  - BCT normally takes zero cycles.
- A resolved branch that follows an unresolved branch causes a delay of up to seven cycles.
- Branches to branches eventually cause problems.
- The compiler optimizes branch usage.
  - BCT is used on the innermost loop that can be controlled by a count.
  - Compares are optimized like any other computation.
  - Condition registers are allocated like other registers.
  - Booleans can be allocated to the condition register.

Branching and Optimization

- Compares produce results that are put into a CR just as arithmetic operations produce results in computational registers.

```
        if a = b  then
          :
        else if a < b  then

  This source code is compiled to the following IL.

        C    CR100=RA,RB
        BE   LABEL_1,CR100
          :
    LABEL_1:
        C    CR100=RA,RB
        BE   LABEL_2,CR100
```

- Optimization can eliminate the second compare.
  It may even be able to move the first compare out of a loop.
- Register allocation will assign CR100 to a real Condition register.
- The compiler will attempt to sandwich other ops between the compare and branch to reduce branch delays.

Branching and Optimization

Optional setting of the CR by computational ops makes this possible.

Compiling to BCT

- BCT is very efficient because the branch decision can be made far in advance.
- Constraints on the use of BCT:
  - There is only one count register.
  - Loading the count register takes one fixed point cycle and four more cycles until a BCT can be done in zero cycles.
    Saving the count register requires a copy to a GPR.
  - The loop termination test must be convertible to:

    Subtract one.

    Compare to zero.

    Branch not zero.
  - Linkage conventions do not require that the count register be preserved.
- Loops are rewritten to use the count register at the same time as strength reduction is done.

17

Other Branch Optimizations

- Many scheduling tricks.
- Idioms involving branches are recognized.

  a<0 ? -a : a

  Becomes:

  abs(a)
- Alternate strategies for switch and case type constructs are recognized to avoid loading the link register.
- The link register need not be saved in "leaf" procedures.
- The link register can often be restored early.
- A procedure that has no functional code after a return can bypass the return by loading the link register before the call and then doing a branch instead of a BAL.
  This is "foliation".
- Control flow can sometimes be reduced.

  if a<b then x = 0;

  else x = 1;

  becomes:

  x = 0;

  if a<b then x = 1;
- Boolean variables can be kept in the condition register.
  Parallel computations may have to be done in the GPRs.