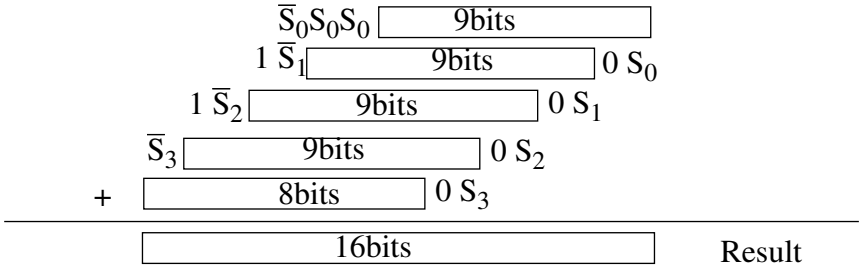
CS 740, Fall 1995 Homework Assignment 3
Sample Answers

Task 1: Multiplier Design Based on Booth Recoding.(30%)

(A) Sign Extension.

The key idea of this justification is that both of original sign extension and the simplified method will contribute the same value to the final result modulus $2n$, where n is the number of bit of our multiplier and multiplicand, because the difference between these two method is the sign extension part.

Let me use the 8x8 multiplier as an example. For each partial product P_i is represented by its sign



bit and n+1 bits of its 1's complement magnitude. Therefore, it can represents the radix-4 partial product in 2's complement form. For n bit multiplier, it will generate $\left\lfloor \frac{n}{2} \right\rfloor + 1$ partial products and the result is the sum of these partial products. For n=32, it generates 17 partial products and the 17th partial product is the multiplicand or zero (always positive). When n=31, it generates 16 partial products and the 16th partial product is 2 times the multiplicand, the multiplicand, or zero (always positive).

Now, let us deal with the sign extension part. For simplification, assume n is even. First, let us sum up the conventional sign extension part:

$$\sum_{i=0}^{\left\lfloor \frac{n}{2} \right\rfloor - 1} \left(S_i \times \sum_{j=n+1+2i}^{2n-1} 2^j \right) = \sum_{i=0}^{\left\lfloor \frac{n}{2} \right\rfloor - 1} S_i \times \left(2^{2n} - 2^{n+1+2i} \right) = \sum_{i=0}^{\left\lfloor \frac{n}{2} \right\rfloor - 1} S_i \times 2^{2n} - \sum_{i=0}^{\left\lfloor \frac{n}{2} \right\rfloor - 1} S_i \times 2^{n+1+2i}$$

Then, we sum up the sign extension part of the modified method and give us this result:

$$2^{n+2\times\left\lfloor\frac{n}{2}\right\rfloor-1}\times\left(1-S_{\left\lfloor\frac{n}{2}\right\rfloor-1}\right)+\sum_{i=1}^{\left\lfloor\frac{n}{2}\right\rfloor-2}(3-S_i)\times 2^{n+1+2i}+(4-S_0)\times 2^{n+1}=\sum_{i=0}^{\left\lfloor\frac{n}{2}\right\rfloor-1}(3-S_i)\times 2^{n+1+2i}-2^{2n}+2^{n+1}$$

$$=2^{2n}-\sum_{i=0}^{\left\lfloor\frac{n}{2}\right\rfloor-1}S_i\times 2^{n+1+2i}$$

For the multiplier, we only consider the first 2n bits, therefore the sums of both sign extension methods mod 2^{2n} will be the same. This proves the correctness of the modified sign extension method, when n is even. Similarly, you can prove the correctness of this modified method, when n is odd.

(B) 32x32 Multiplier Based Radix-4

There are many ways to implement this sequential multiplier. The Bit vector version is shown as mult_vect.c. In gcc, you can use “long long” to represent 64-bit integers and “long long unsigned” to represent 64-bit unsigned integer. A “long long” version is shown as mult_long.c. This latter version is much short that the former one(200 line shorter).

Task 2: Floating-Point Rounding(30%)

Assume n is the number bits to represent the mantissa part and m is the value of mantissa, $1.0 \le m \le 2.0 - 2^{-n}$. Let M_a and M_b represents the mantissa parts of a and b respectively. The exactness of the result of a+b or a-b is determined by whether the n bit of mantissa part can exactly represent the mantissa part of a+b or a-b.

(A) Since a and b have the same exponents, there will be no right shift before addition. The signs of a and b are different, therefore it will perform the true subtraction on the mantissa parts of a and b, the result will be less than 2 that implies there is no right shift after addition. So the result is exact. In other words, the shifting will cause that the guard bit (g), round bit (r) and sticky bits (s) probably become nonzero, therefore the right shift on the mantissa part before or after the addition will cause that the result isn’t exact.

(B) The exponents of a and b differ by 1, therefore either a or b will be shifted right one bit before addition. Since the signs of a and b are different, it will perform true subtraction on M_a and M_b. Assume M_b is shifted right by 1 bit. If the least significant bit of M_b is 1, then it need n+1 bit to represent the result of mantissa part after addition (actually true subtraction M_a-M_b). Therefore, the result of a+b isn’t exact. If the least significant bit of M_b is 0, then the n+1th bit of the result is 0 and the result of $M_a + M_b$ can be exactly represented by n bits. Similarly, you can show the same properties, when M_a is shifted right by 1.

(C) Assume $a \geq b \geq 0$, and the top two bits of a cancel when computing a-b. Let M_{bs} represents M_b right shifted by s bits. The top two bits of M_a - M_{bs} are zero, so M_a -M_{bs} < 0.5. We know that $1.0 \leq M_a \leq 2.0 - 2^{-n}$ and $1.0 \leq M_b \leq 2.0 + 2^{-n}$. If M_b is right shifted by more than 1 bits, the M_{bs} will be less than 0.5 and the top two bits of the result couldn't be all zeros (M_a -M_{bs}>0.5). Therefore, Mb can be only right shifted by 1 or 0 bit. By assumption, the top two bits of the result are zero, so the result must be left shifted by at least 2 bits. Therefore, the effect of the exactness of the result caused by the right shift by at most 1 bit before addition is cancelled by the left shift at least 2 bits of the result. In other words, the result is exact.

(D) Assume $a \geq b \geq 0$ and the exponents differ by 1. This implies that M_b will be righted by 1 bit. M_a - (M_b right shifted by 1 bit) can be represented in n+1 bits. If the high bit of the result is zero, then the result will be left shifted by 1 bit and can be represented in n bits. If the high bit of result is 1, the there is no left shift on the result and the result can't be exactly represented in n bits.

(E) Assume the result of a-b or a+b is denormal. Here you can't assume that $a, b \geq 0$. If you do that, you only did part of this question. It will cost your 2 points. Let E is the exponent value of denormal representation. The finest precision of floating point representation is 2^{E-n}. The result of any addition or subtraction is always a multiple of this finest value. If the result is denormal, no right shift is needed. Therefore the result is exact.

Task 3: Cache Measurements (40%)

This task is related to the machine you choose, therefore there is no standard answer for this questions. Here are the methods used to measure these cache parameters on DEC 5000/20 personal.

1. Physical or virtual addressing:

You have to measure the equivalent class of a lot number (64 or 128) of pages to see whether there is a regular patter to form the equivalent classes. If it has a regular patter, then it is virtual addressing. Otherwise, it is physical addressing. It is physical addressing on DEC 5000/20 personal. The program is in asst3/cache_equ.c file. If the page size is the same as the cache size, It will be very difficult to do this measurement.

2. Other parameters:

The other parameters can be measure by asst3/cache_test.c program. The following are the measurement on DEC 5000/20 Personal.

Determine line size:

Offset = 1,Cycles/load = 25.833235 * Miss 1/4 time

Offset = 2,Cycles/load = 32.747864 * Miss 1/2 time

Offset = 4,Cycles/load = 45.310600 * Miss every time

Offset = 8,Cycles/load = 45.548073

Offset = 16,Cycles/load = 45.347855

We can conclude that the line size is 4 words (16 bytes) and the miss penalty is around 26 cycles.

Determine associativity :

Items = 1,Cycles/load = 30.605957

Items = 2,Cycles/load = 55.303047 * Penalty Visible, Direct-Mapped.

Items = 3,Cycles/load = 54.558037

Items = 4,Cycles/load = 54.413696

Items = 5,Cycles/load = 54.092407

Items = 6,Cycles/load = 54.246067

Items = 7,Cycles/load = 53.845627 I

tems = 8,Cycles/load = 53.743187

We can conclude that the cache is direct-map and no replacement policy.

Assuming line size of 4 words ...

Determine total size

Range = 512,Cycles/load = 19.658962

Range = 1024,Cycles/load = 19.486677

Range = 2048,Cycles/load = 19.398209

Range = 4096,Cycles/load = 19.416832 * Complete in Cache

Range = 8192,Cycles/load = 25.949642 * Degrading the performance

Range = 16384,Cycles/load = 26.378023

Range = 32768,Cycles/load = 43.601730 * Degarding complete.

Range = 65536,Cycles/load = 45.245407

Range = 131072,Cycles/load = 45.543411

Range = 262144,Cycles/load = 45.324570

We can conclude that the cache size is 64KB(4K x 4W).

Grade Distributions:

100---90 : 11

89 ---80 : 5

79 ---70 : 1

69 ---60 : 1