System programming assignment2

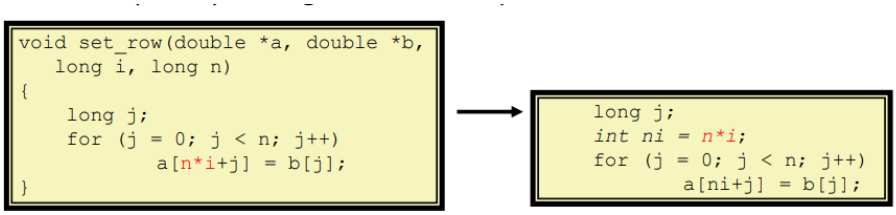
202211162

First section (Test result in rasberrypi)

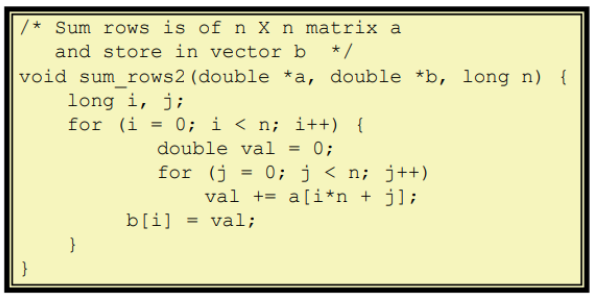
	Img_128.bmp	Img_256.bmp	Img_512.bmp	Img_1024.bmp
Speed up(times)	1.732392	1.740919	1.739897	2.058095

Second section:

I didn't declare a convolution function for compiler optimization. I used the following strategies. Reduction in strength, Removing aliasing, Loop unrolling. Except of loop unrolling, each strategy has increased the speed up about 20% (+0.2). and loop unrolling does about 30%. Reduction in strength is extracting the value that has already been calculated out of the 'for' statement. It can reduce the calculation. Removing aliasing is making a memory access faster by not accessing repetitive array pointer. It extracts array out of 'for' statement. In Loop unrolling, it is to calculate multiple additions in one 'for' statement. It can reduce a number of looping. I set unrolling factor L is 3, K is 3. When L, K > 3, It didn't increase speed. The Figures below illustrate how to apply the strategies (from extra lecture note).



Reduction in strength



Removing aliasing

```
void combine4(vec_ptr v, data_t *dest)
{
  int i;
  int length = vec_length(v);
  data_t *d = get_vec_start(v);
  data_t t = IDENT;
  for (i = 0; i < length; i++)
    t = t OP d[i];
  *dest = t;
}
```

→

```
void unroll2aa_combine(vec_ptr v, data_t *dest)
{
    int length = vec_length(v);
    int limit = length-1;
    data_t *d = get_vec_start(v);
    data_t x = IDENT;
    int i;
    /* Combine 2 elements at a time */
    for (i = 0; i < limit; i+=2) {
        x = x OP (d[i] OP d[i+1]);
    }
    /* Finish any remaining elements */
    for (; i < length; i++) {
        x = x OP d[i];
    }
    *dest = x;
}
```

Compare to before

```
x = (x OP d[i]
```

Loop unrolling.