CS 4400 - Problem Set 6

Rob Johansen

u0531837

---

1. Problem 3.67:

   A. e1.p is at byte offset 0
      e1.y is at byte offset 4
      e2.x is at byte offset 0
      e2.next is at byte offset 4

   B. The total bytes required for this union would be 8.

   C. Here is the missing code for the proc() function:

```
void proc(union ele *up)
{
    up->next.y = *(up->next.p) - up->x;
}
```

2. Problem 5.16. This is a version that uses four-way loop unrolling. Code that is new or modified has been highlighted in red:

```
void inner4(vec_ptr u, vec_ptr v, data_t *dest)
{
    long int i;
    int length = vec_length(u);
    int limit  = length - 3;
    data_t *udata = get_vec_start(u);
    data_t *vdata = get_vec_start(v);
    data_t sum = (data) 0;
    for (i = 0; i < limit; i += 4) {
        sum = (((sum + udata[i]   * vdata[i])
                     + udata[i+1] * vdata[i+1])
                     + udata[i+2] * vdata[i+2])
                     + udata[i+3] * vdata[i+3];
    }
    for (; i < length; i++) {
        sum = sum + udata[i] * vdata[i];
    }
    *dest = sum;
}
```

A. No version can achieve a CPE less than 2.00 because of the data-dependency chain between loop registers.

B. Performance did not improve for loop unrolling because there are still *n* multiply operations. It's true that there are now 1/4 as many iterations, but each iteration now has 4 multiplications in sequence.

3. Problem 5.17. This is a version that uses four-way loop unrolling with four parallel accumulator. Code that is new or modified has been highlighted in red:

```
void inner4(vec_ptr u, vec_ptr v, data_t *dest)
{
      long int i;
      int length = vec_length(u);
      int limit  = length - 3;
      data_t *udata = get_vec_start(u);
      data_t *vdata = get_vec_start(v);
      data_t sum0 = (data) 0;
      data_t sum1 = (data) 0;
      data_t sum2 = (data) 0;
      data_t sum3 = (data) 0;

      for (i = 0; i < limit; i += 4) {
            sum0 = sum0 + udata[i]   * vdata[i];
            sum1 = sum1 + udata[i+1] * vdata[i+1];
            sum2 = sum2 + udata[i+2] * vdata[i+2];
            sum3 = sum3 + udata[i+3] * vdata[i+3];
      }

      for (; i < length; i++) {
            sum0 = sum0 + udata[i] * vdata[i];
      }
      *dest = sum0 + sum1 + sum2 + sum3;
}
```

A. The performance is limited to a CPE of 2.00 because of the data-dependency chain between loop registers.

B. The version with integer data on IA32 is worse because a new data-dependency chain between loop registers is introduced.

4. Problem 5.18. This is a version that uses four-way loop unrolling along with reassociation. Code that is new or modified has been highlighted in red:

```
void inner4(vec_ptr u, vec_ptr v, data_t *dest)
{
      long int i;
      int length = vec_length(u);
      int limit  = length - 3;
      data_t *udata = get_vec_start(u);
      data_t *vdata = get_vec_start(v);
      data_t sum = (data) 0;

      for (i = 0; i < limit; i += 4) {
            sum = sum + ((((udata[i]   * vdata[i])
                        + udata[i+1] * vdata[i+1])
                        + udata[i+2] * vdata[i+2])
                        + udata[i+3] * vdata[i+3]);
      }

      for (; i < length; i++) {
            sum = sum + udata[i] * vdata[i];
      }
      *dest = sum;
}
```