Problem I

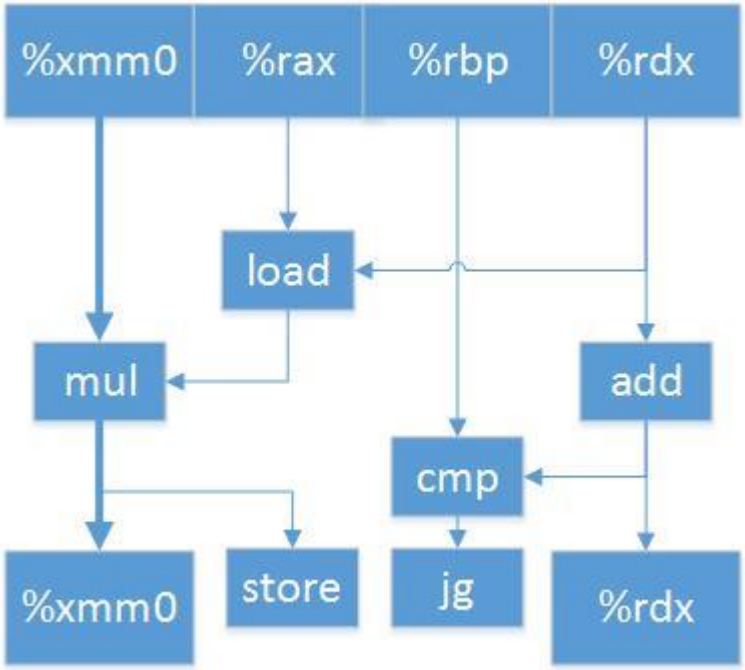
The assembly code generated for the compiled loop of combine3 with optimization level 2 is shown below:

combine3: data_t = float, OP = *, compiled –O2

i in %rdx, data in %rax, limit in %rbp, dest at %rx12, product in %xmm0

```
1 .L560:                                loop:
2 mulss   (%rax, %rdx, 4), %xmm0        Multiply product by data[i]
3 addq    $1, %rdx                      Increment i
4 cmpq    %rdx, %rbp                    Compare limit : i
5 movss   %xmm0, (%r12)                 Store product at dest
6 jg      .L560                         If >, go loop
```

Illustrate the code above with **data-flow graph** like figure 5.14(a) or (b) in textbook and mark the **critical path**. You can use "store" to identify operation in line 5



Problem II

The following code for PrefixSum seems not very good. Please rewrite it

```
/*Compute prefix sum of vector v*/
1   void prefixSum(float v[], float p[], long int n)
2   {
3       long int i;
4       long int limit = n-2;
5       /* acc is accumulator; */
6       float acc; = 0
7       for ( i = 0; i < limit; i+=3 )
8       {
9           acc += v[i]
10          p[i] = acc;
11          acc += v[i+1];
12          p[i+1] = acc;
13          acc += v[i+2];
14          p[i+2] = acc;
15      }
16      for (; i < n; i++)
17      {
18          acc = acc + v[i];
19          p[i] = acc;
20      }
21  }
```

Problem III

Suppose an allocator has a heap with size of **(5 * SIZE)**, and the heap is **empty** initially. Then, a user makes a sequence of requests as shown in the table below.

a). What's the aggregate payload after the execution of code in line 7? What's the peak memory utilization after these requests?

```
1.  void *pa, *pb, *pc;
2.
3.  pa = malloc(SIZE);
```

```
4.  free(pa)
5.  pa = malloc(SIZE);
6.  pb = malloc(SIZE);
7.  free(pa)
8.  pa = malloc(SIZE);
9.  pc = malloc(SIZE);
10. free(pa);
11. free(pb);
12. free(pc);
```

Because pa is freed, the aggregate payload only consists of pb. So, aggregate payload is SIZE.

Peak memory utilization is 3 * SIZE / 5 * SIZE = 0.6.

b). How do we know how much memory to free just given a pointer?

By keeping the length of a structure in the word preceding the allocated structure.