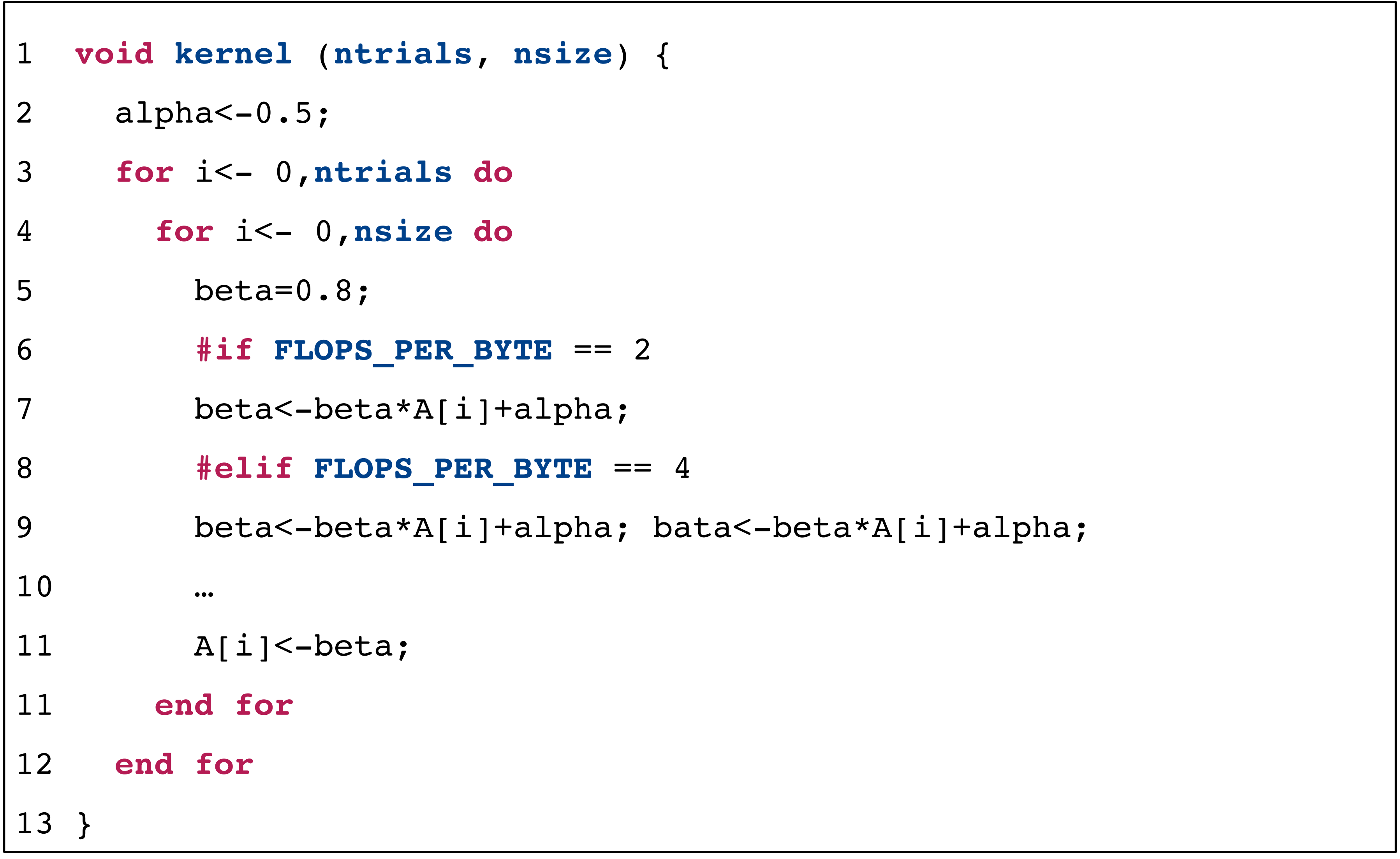
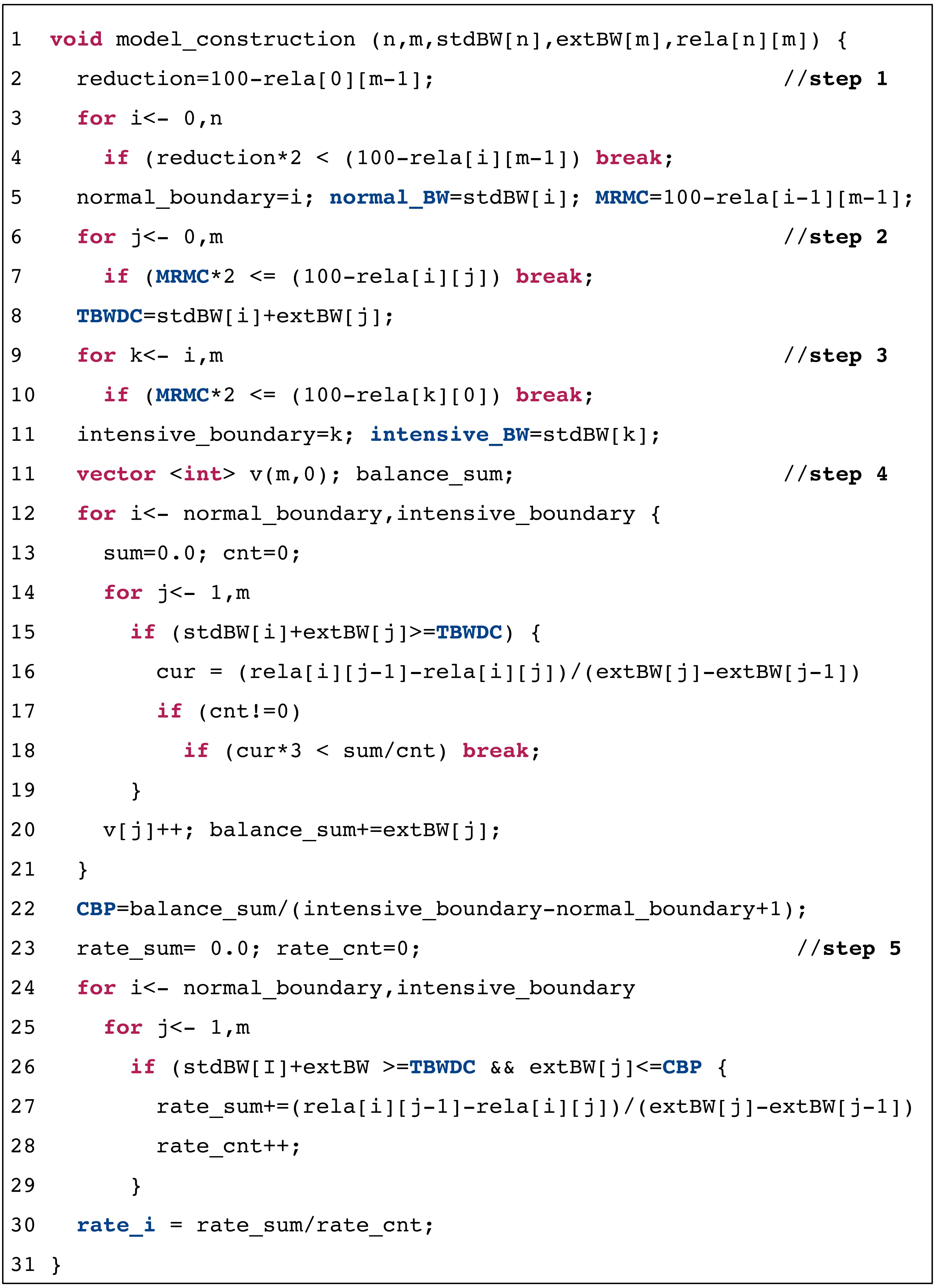


```
void kernel (ntrials, nsize) {
  alpha<-0.5;
  for i<- 0,ntrials do
    for i<- 0,nsize do
      beta=0.8;
      #if FLOPS_PER_BYTE == 2
      beta<-beta*A[i]+alpha;
      #elif FLOPS_PER_BYTE == 4
      beta<-beta*A[i]+alpha; bata<-beta*A[i]+alpha;
      …
      A[i]<-beta;
    end for
  end for
}
```

(a) The kernel to generate different BW by varying compute/memory (i.e., operational) intensities



```
void model_construction (n,m,stdBW[n],extBW[m],rela[n][m]) {
  reduction=100-rela[0][m-1];                          //step 1
  for i<- 0,n
    if (reduction*2 < (100-rela[i][m-1]) break;
  normal_boundary=i; normal_BW=stdBW[i]; MRMC=100-rela[i-1][m-1];
  for j<- 0,m                                          //step 2
    if (MRMC*2 <= (100-rela[i][j]) break;
  TBWDC=stdBW[i]+extBW[j];
  for k<- i,m                                          //step 3
    if (MRMC*2 <= (100-rela[k][0]) break;
  intensive_boundary=k; intensive_BW=stdBW[k];
  vector <int> v(m,0); balance_sum;                    //step 4
  for i<- normal_boundary,intensive_boundary {
    sum=0.0; cnt=0;
    for j<- 1,m
      if (stdBW[i]+extBW[j]>=TBWDC) {
        cur = (rela[i][j-1]-rela[i][j])/(extBW[j]-extBW[j-1])
        if (cnt!=0)
          if (cur*3 < sum/cnt) break;
      }
    v[j]++; balance_sum+=extBW[j];
  }
  CBP=balance_sum/(intensive_boundary-normal_boundary+1);
  rate_sum= 0.0; rate_cnt=0;                            //step 5
  for i<- normal_boundary,intensive_boundary
    for j<- 1,m
      if (stdBW[I]+extBW >=TBWDC && extBW[j]<=CBP {
        rate_sum+=(rela[i][j-1]-rela[i][j])/(extBW[j]-extBW[j-1])
        rate_cnt++;
      }
  rate_i = rate_sum/rate_cnt;
}
```

(b) The model construction