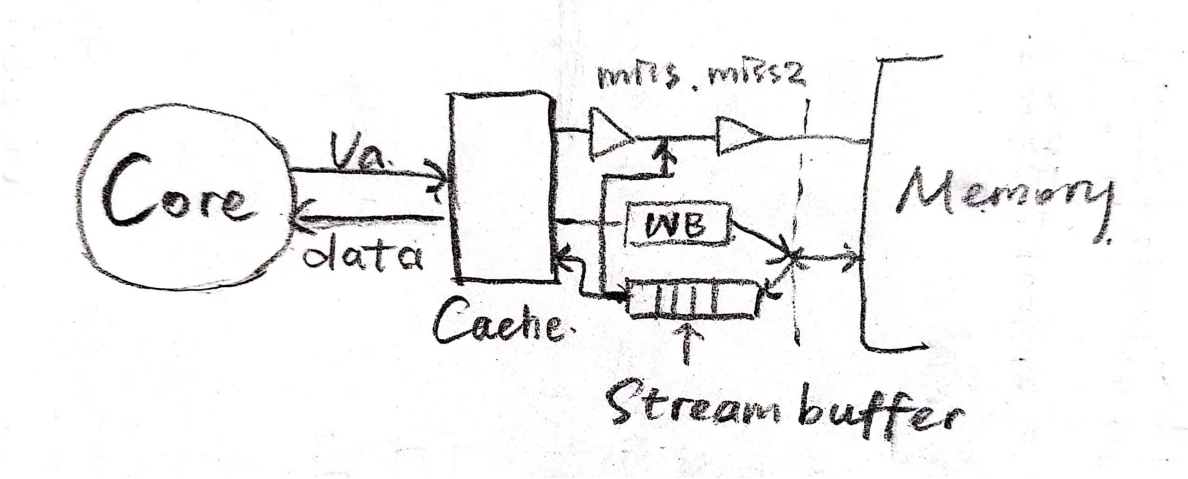
Note W7D1

xjq

核心内容：由于 $AMAT = T_{hit} + \eta \times T_{penalty}$，因此我们希望降低 η 来减少 $AMAT$。

- Hardware prefetching



第一次miss之后先去stream buffer找（多用于数组的访问

- Software Prefetching
- Compiler opt

① merging arrays

例如：

```
int key[];
int val[];
////////////////
struct{
    int key,val;
}mergeArray[];
```

大多数情况下，key和val会同时访问，那么这个时候同一对放在一起存储访问效率会更高。

② loop interchange

例如：

```
for j
    for i
        x[i][j]
/////////////
for i
    for j
        x[i][j]
```

存储方式是按照每个维数的下标按顺序的字典序来依次存储的，所以先枚举 i 再枚举 j 会更快。

③ loop fusion

合并循环

④ blocking

例如：

```
for i
    for j
        for k
            a[i][j]=a[i][j]+b[i][k]*c[k][j]
```

我们可以考虑对 i, j 进行分块：

```
for (jj = 0; jj < N; jj = jj+B)
for (kk = 0; kk < N; kk = kk+B)
for (i = 0; i < N; i = i+1)
for (j = jj; j < min(jj+B-1,N); j = j+1){
    r = 0;
    for (k = kk; k < min(kk+B-1,N); k = k+1) {
        r = r + b[i][k]*c[k][j];
    }
    a[i][j] = a[i][j] + r;
}
```

这样使得 (jj, kk) 所在的块很有可能一直在cache里（设置块的大小时和cache的大小有关?