```
typedef long long int VA;
typedef long long int PA;
typedef long long int VPN;
typedef long long int PPN;
typedef long long int OFFSET;

VPN getVPN(VA);                                     /* Extracts the first 52 bits of a VA */
OFFSET getOffset(VA);                               /* Extracts the last 12 bits of a VA */
PA concat(PPN, OFFSET);                             /* Concatenate PPN and OFFSET to produce a PA */

class Page {};                                      /* Page: a block in virtual memory system */
class Data {};                                      /* Data: 32-bit data stored inside RAM index by a physical address */

class _TLB {                                        /* Translation Lookaside Buffer */
    struct TLBRow { int validBit; int dirtyBit; int refBit; VPN tag; PPN ppn; };
    vector<TLBRow> table;
public:
    bool hit(VPN vpn);                              /* Returns true if given vpn exists in TLB */
    PPN get(VPN vpn);                               /* Returns the corresponding PPN of the given VPN */
    void add(VPN vpn, PPN ppn);                     /* Adds a new entry to the table, removes entry if full */
};

class _PageTable {                                  /* Page Table inside RAM */
    struct PageTableEntry { int validBit; int dirtyBit; int refBit; PPN ppn; };
    vector<PageTableEntry> table;
public:
    bool hit(VPN vpn);                              /* Returns true if given vpn exists in TLB */
    PPN get(VPN vpn);                               /* Returns the corresponding PPN of the given VPN */
    void add(VPN vpn, PPN ppn);                     /* Adds a new entry to the table, removes entry if full */
};

struct _RAM {
    void load(Page page);                           /* Load a page into RAM */
    PPN locate(Page page);                          /* Locates PPN on RAM */
    Data getData(PA pa);                            /* Returns M[pa], the data stored at physical address pa */
}

struct _DISK {
    Page get(VPN vpn);                              /* Extract page given virtual page number */
}

struct _Cache {
    bool hit(PA pa);                                /* Returns true if given pa is cached */
    Data get(PA pa);                                /* Extract data given physical address */
}

/* Globals */
_TLB TLB;                                        /* Our TLB */
_PageTable PageTable;                            /* Our PageTable */
_RAM RAM;                                        /* Our RAM */
_DISK DISK;                                      /* Our Disk */
_Cache CACHE;                                    /* Our Cache */

/* Given a virtual address, return the corresponding physical address */
PA translation (VA va) {                           /* va.length() = 64; */

    VPN vpn = getVPN(va);                           /* vpn.length() = 52; */
    OFFSET offset = getOffset(va);                  /* offset.length() = 12; */
    PPN ppn;                                        /* ppn.length() = 20 */
    PA pa;                                          /* pa.length() = 32 */

    if (TLB.hit(vpn)) {                             /* If TLB hits */
        ppn = TLB.get(vpn);                         /* Get the corresponding PPN */
    }
    else {                                          /* If TLB misses */

        if (PageTable.hit(vpn)) {                   /* If Page Table hits */
            ppn = PageTable.get(vpn);               /* Get the corresponding PPN */
            TLB.add(vpn, ppn);                      /* Updates TLB, removes entry if necessary */
        }
        else {                                      /* If Page Table misses */
            Page p = DISK.get(vpn);                 /* Find page in disk using vpn */
            RAM.load(p);                            /* Load page into RAM */
            ppn = RAM.locate(p);                    /* Locate the recently-loaded page on RAM */
            PageTable.add(vpn, ppn);                /* Update page table, removes entry if necessary */
            TLB.add(vpn, ppn);                      /* Update TLB, removes entry if necessary */
        }
    }
    pa = concatenate(ppn,offset);                   /* Finally, we get the physical address */
    return pa;                                      /* Return the physical address we got */
}

/* Given a physical address, find its data */
Data getData(PA pa) {
    Data result;                                    /* Our output */

    if (Cache.hit(pa)) {                            /* If PA is cached */
        result = Cache.get(pa);                     /* Extract the cached data */
    }
    else {                                          /* If PA is not cached */
        result = RAM.getData();                     /* Load the data from RAM */
        cache.add(pa, result);                      /* Update Cache, remove entry if necessary */
    }
    return result;                                  /* Return the data we got */
}
```