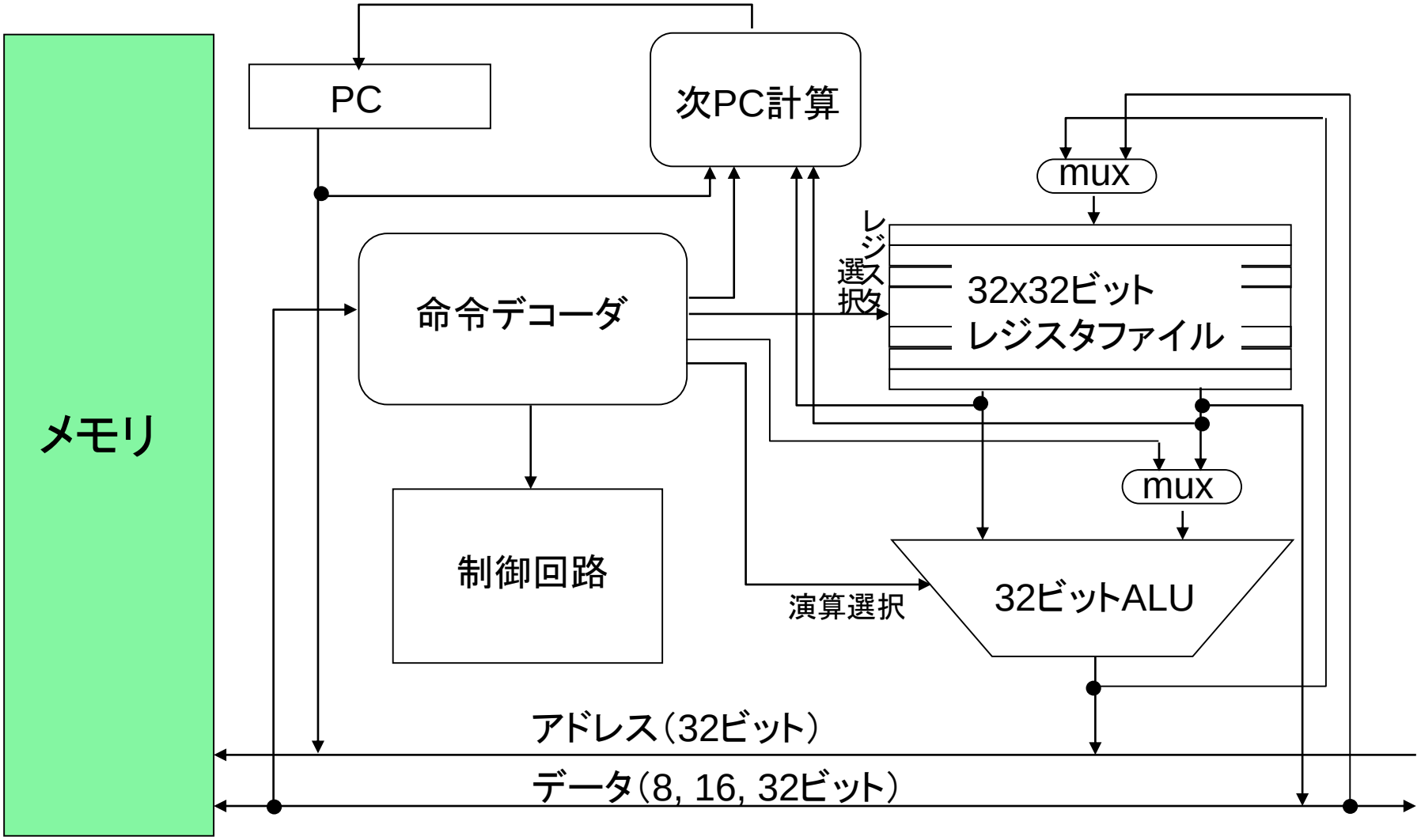
東北大学 工学部 機械知能･航空工学科
2019年度 クラス C D

情報科学基礎 I

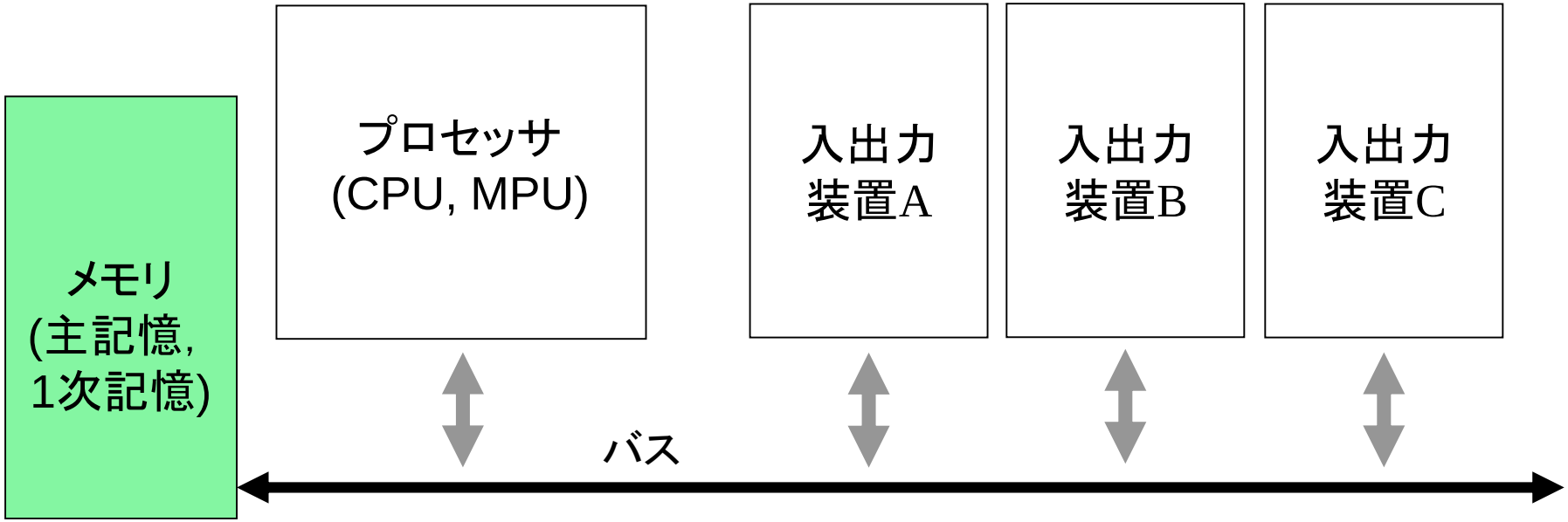
13. メモリシステム
(教科書8章)

大学院情報科学研究科
鏡 慎吾
http://www.ic.is.tohoku.ac.jp/~swk/lecture/

(復習) MIPSの構造


メモリ
PC
次PC計算
mux
レジ選択
32x32ビット
レジスタファイル
命令デコーダ
mux
制御回路
演算選択
32ビットALU
アドレス（32ビット）
データ（8, 16, 32ビット）

(復習) 計算機の基本構成



別物!

入出力装置 (Input/Output, I/O) の例

- 二次記憶（外部記憶，ストレージ）: ハードディスク, CD, DVD
- キーボード，マウス
- グラフィックス，ディスプレイ
- ネットワーク

「メモリ」という用語の混乱

Q: 以下の会話は，2017年頃のウェブ上で実際に見られたやり取りの例である（一部改変）．何がおかしいのかを指摘せよ．

「当社では社員が使う PC のメモリはすべて 32 GB です．快適に作業ができます」
「えっ? 32 GB って少なくないですか? 私の iPhone のメモリは 128 GB なんですけど」

A: 前者は主記憶の話をしている．後者は二次記憶の話をしている

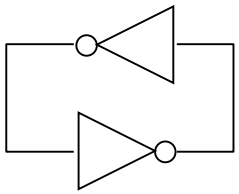
- アーキテクチャの観点では，主記憶をメモリ，二次記憶をストレージと呼ぶことが多い
- デバイスの観点では，半導体記憶素子をすべてメモリと呼んでしまうことがある (e.g.:「フラッシュメモリ」は半導体記憶素子だが，主な用途は二次記憶)
- そこにマーケティング上の思惑が絡むのでさらにややこしい，というか迷惑

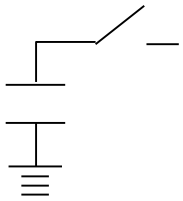
Cプログラムの場合

```
int main() {
    FILE *fp;
    char str[1024];

    fp = fopen("file.txt");
    fgets(str, 1024, fp);
    ...
}
```

- プログラム上の変数は（普通は）主記憶上にある
  - load/store命令で直接読み書きされる
- 二次記憶上のデータは，入出力関数を使って読み書きする
  - 専用の入出力命令を使って機器の制御が行われる
    - プロセッサによっては，特定の主記憶アドレスへのload/store によって入出力制御を行うものもある (例: MIPS)

記憶装置の原理





- フリップフロップ
  - レジスタ
  - SRAM → キャッシュメモリ（後述）
- キャパシタ
  - DRAM → 主記憶
- 磁気
  - 磁気記憶装置 (ハードディスク)
    → 二次記憶
  - 最近はフラッシュメモリによる置き換えが進んでいる

揮発性
（電源を切ると内容は消える）

不揮発性

速

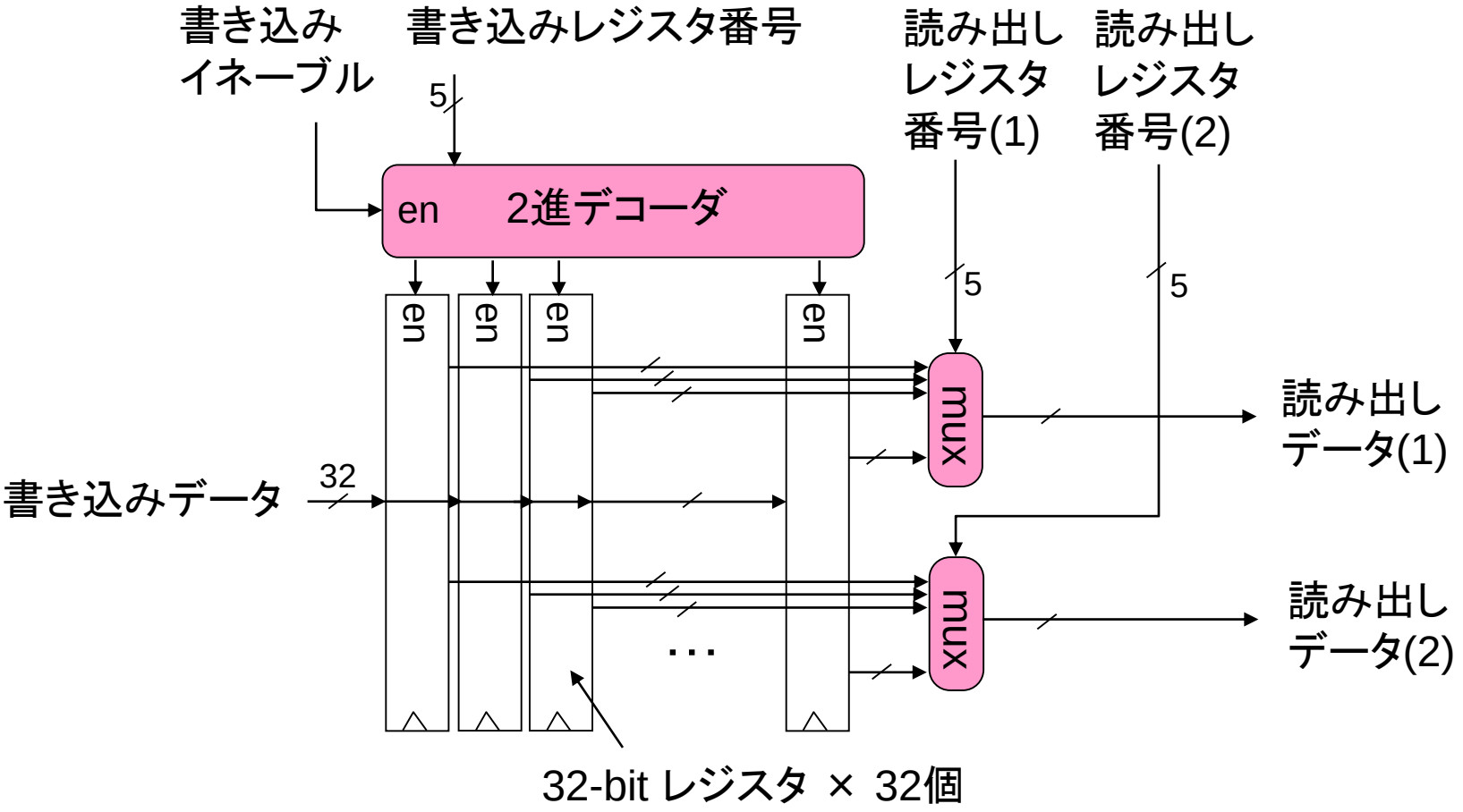
遅

https://commons.wikimedia.org/w/index.php?title=File:HardDisk1.ogv

ハードディスクの動作

https://commons.wikimedia.org/w/index.php?title=File:HardDisk1.ogv

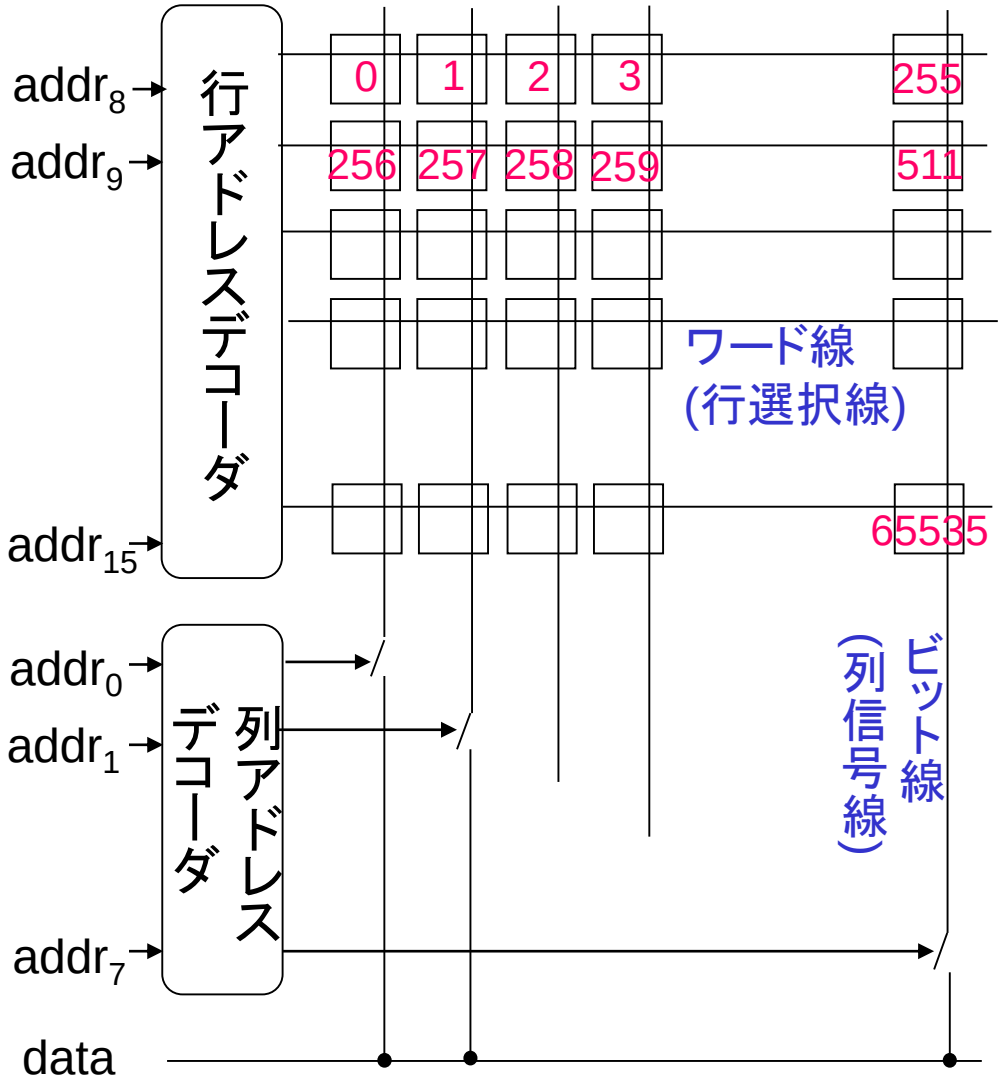
#fis1 (復習) レジスタファイル (32×32ビット，1入力2出力)



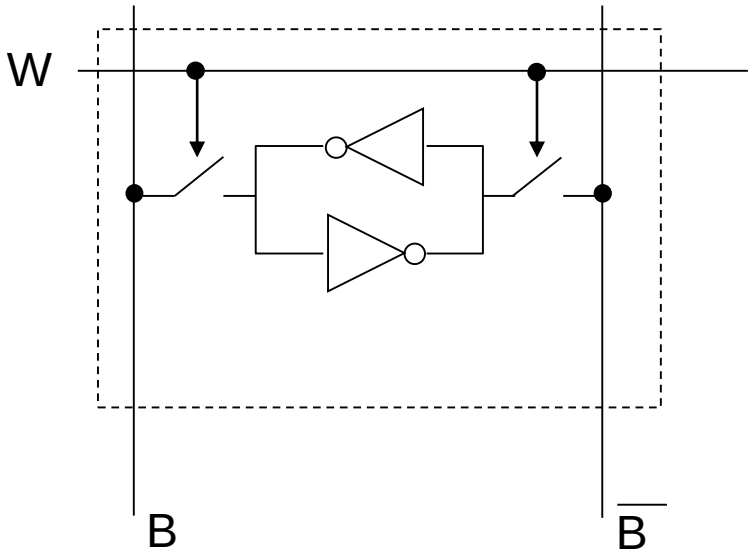
この構造のまま，単純に容量を拡大するのは困難
（デコーダ・マルチプレクサが肥大化するため）

半導体メモリの構造

- セル（記憶素子）を2次元マトリックス状に配置して，アクセスを縦・横に分解
- 読み出し・書き込み回路を共通化

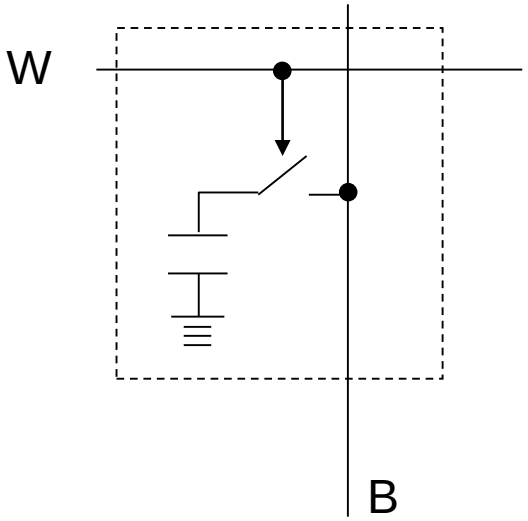


メモリセルの構造



Static Random Access Memory (SRAM)

- フリップフロップと同じく2個のNOTゲートのループ構成により記憶
- NOTゲートが能動的に電流を供給してビット線を駆動する
- 1ビットあたりトランジスタ 6 個



Dynamic Random Access Memory (DRAM)

- キャパシタが充電されていれば 1，放電されていれば 0
- ビット線の電位は，キャパシタからのわずかな電荷で微小に変化する
- 時間とともに電荷が漏れる
- 1ビットあたりトランジスタ 1 個

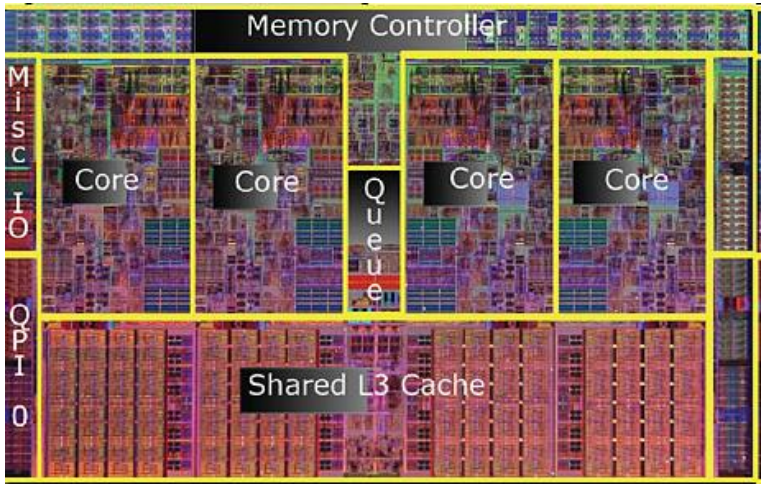
DRAM と SRAM

DRAMモジュール

http://www.sugilab.net/jk/joho-kiki/

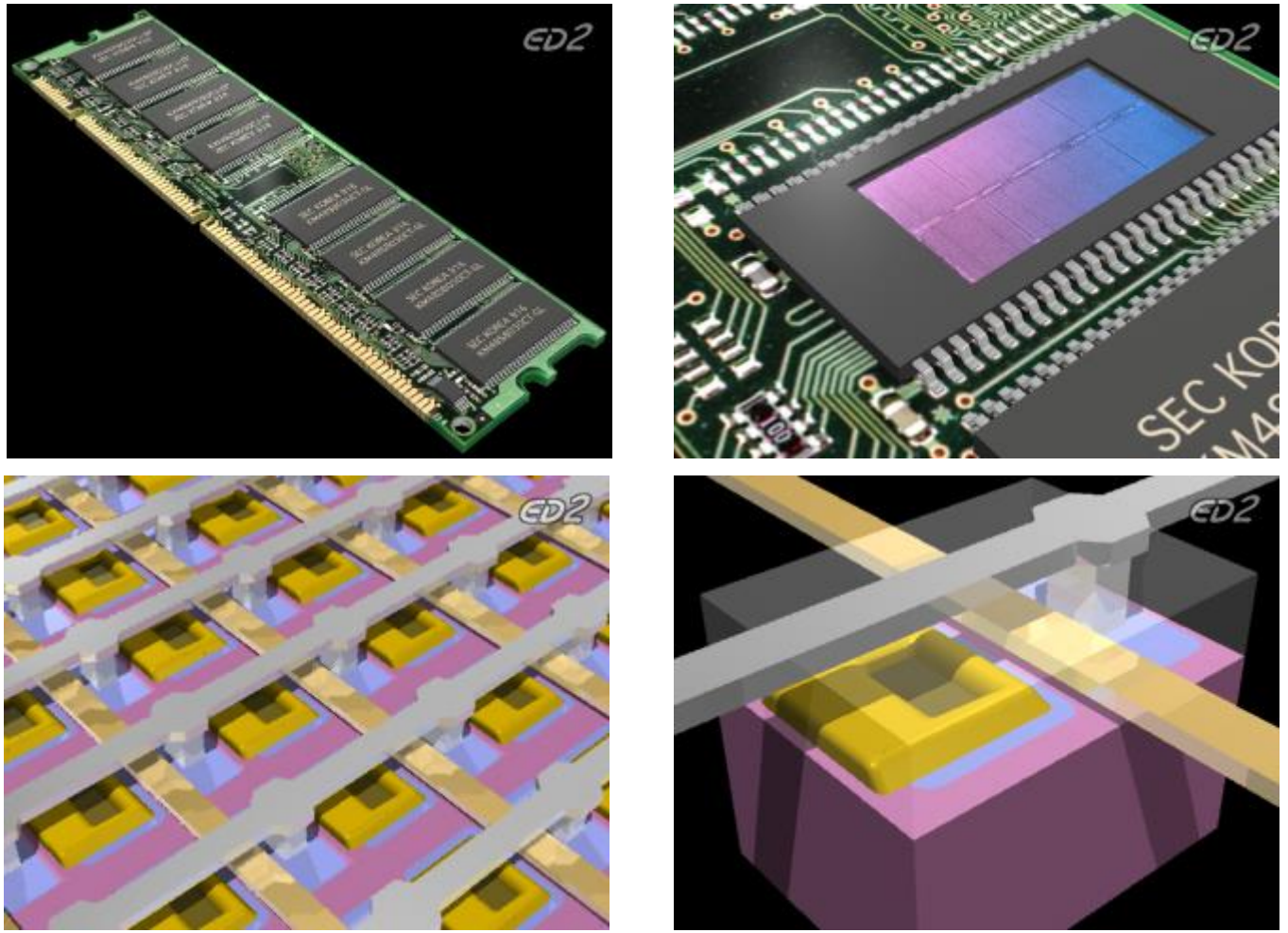
SRAM (キャッシュメモリ) 内蔵プロセッサ

http://ja.wikipedia.org/wiki/Intel_Core_i7



http://www.atmarkit.co.jp/fsys/zunouhoudan/102zunou/corei7.html

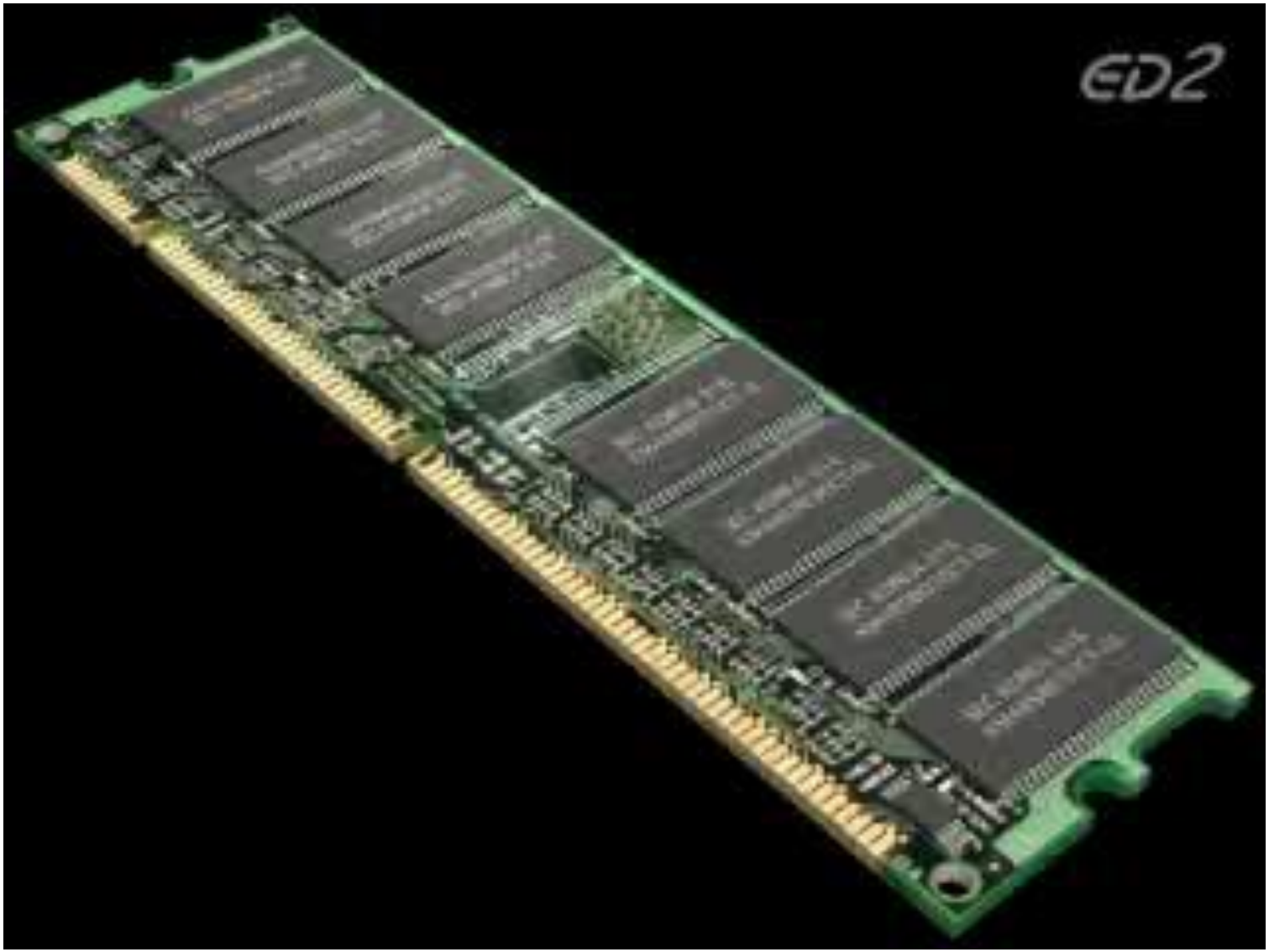
メモリの動作 (DRAMの場合)



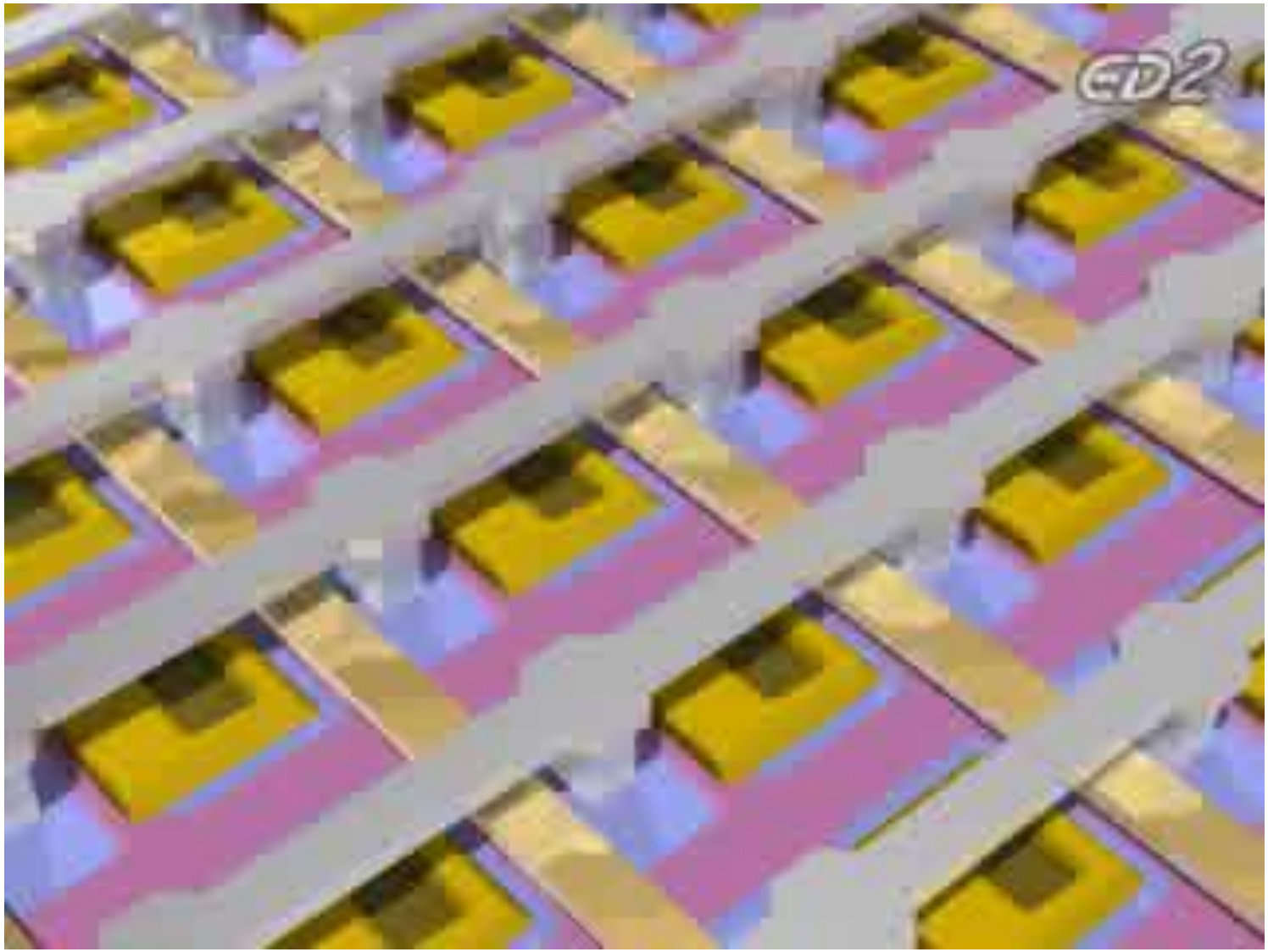
http://www.sugilab.net/jk/joho-kiki/

(1400 処理装置 → 1404 メモリのしくみ)

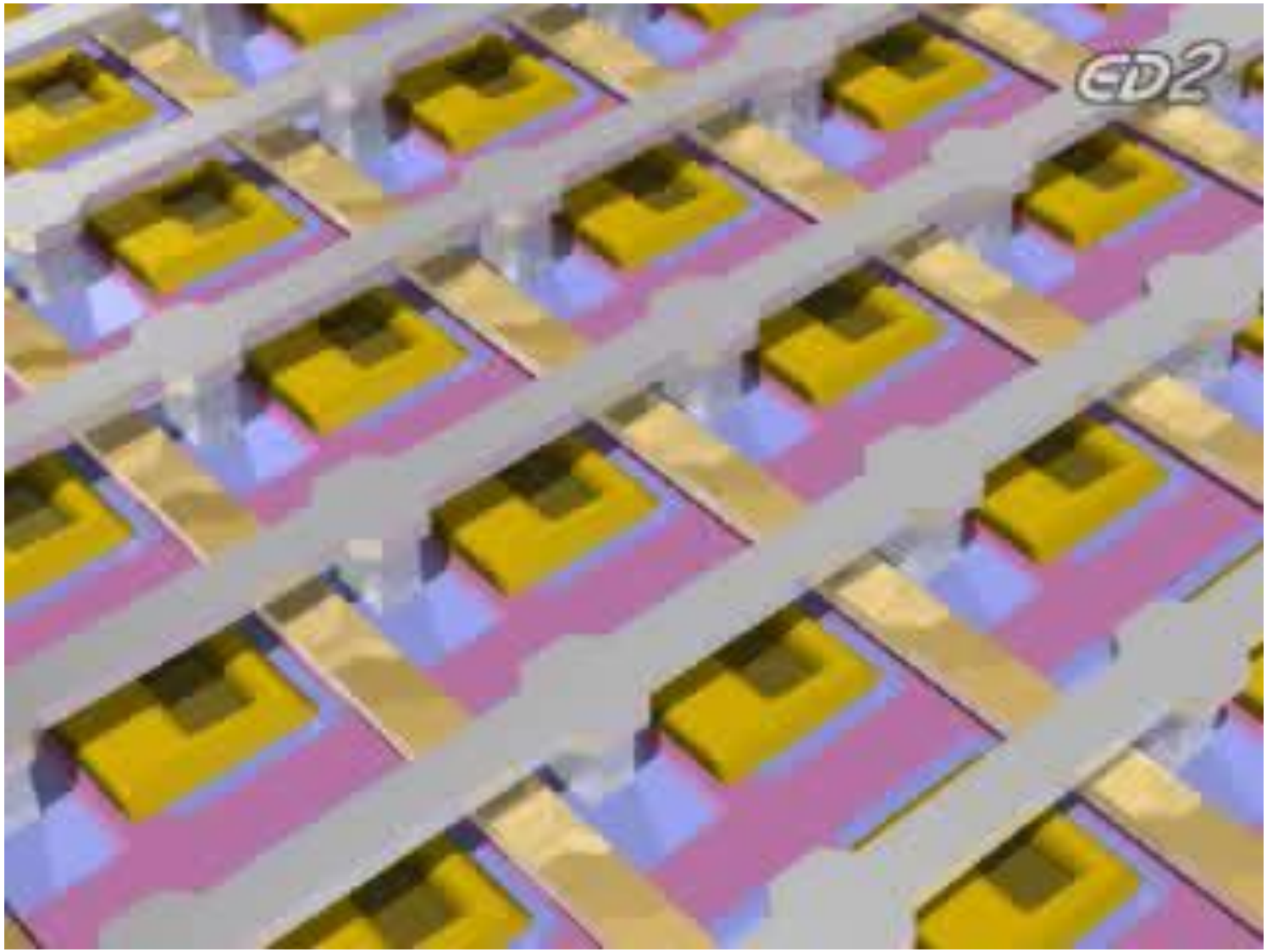
構造



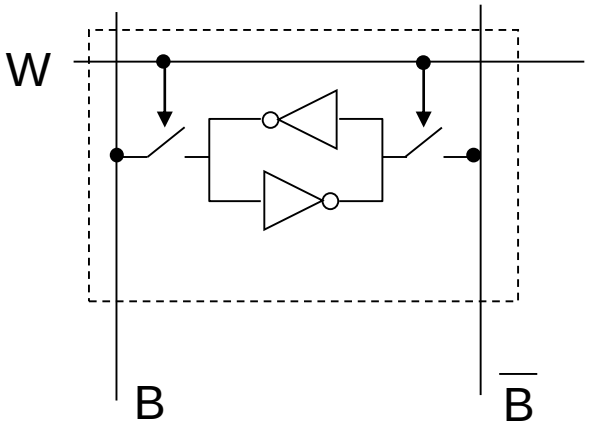
書き込み



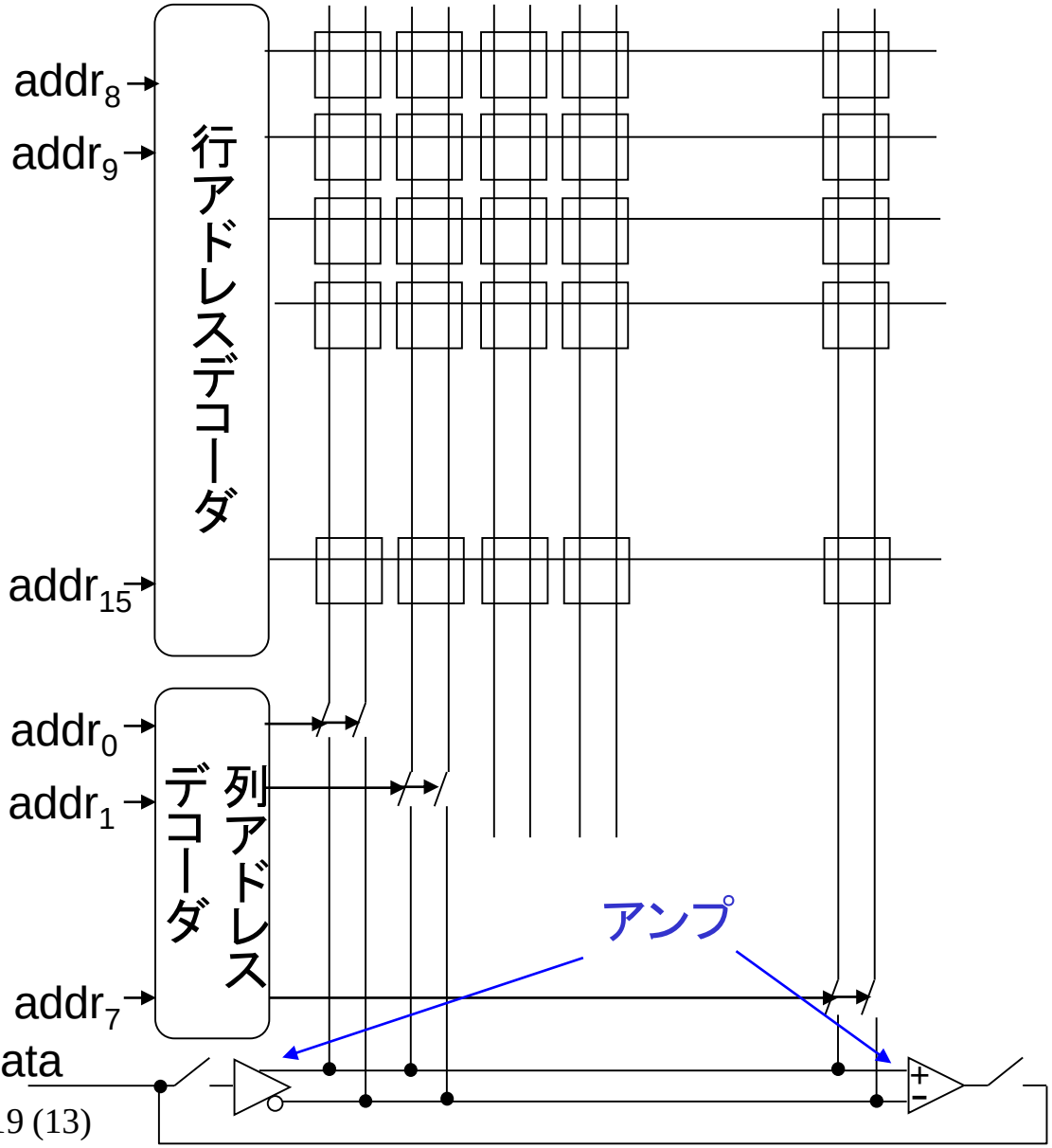
読み出し



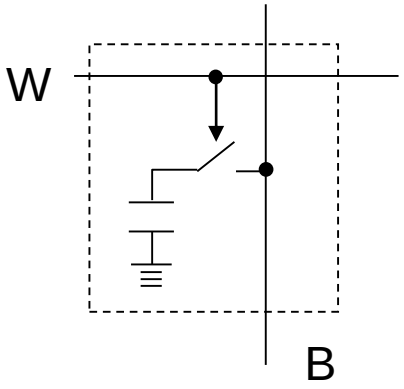
SRAM



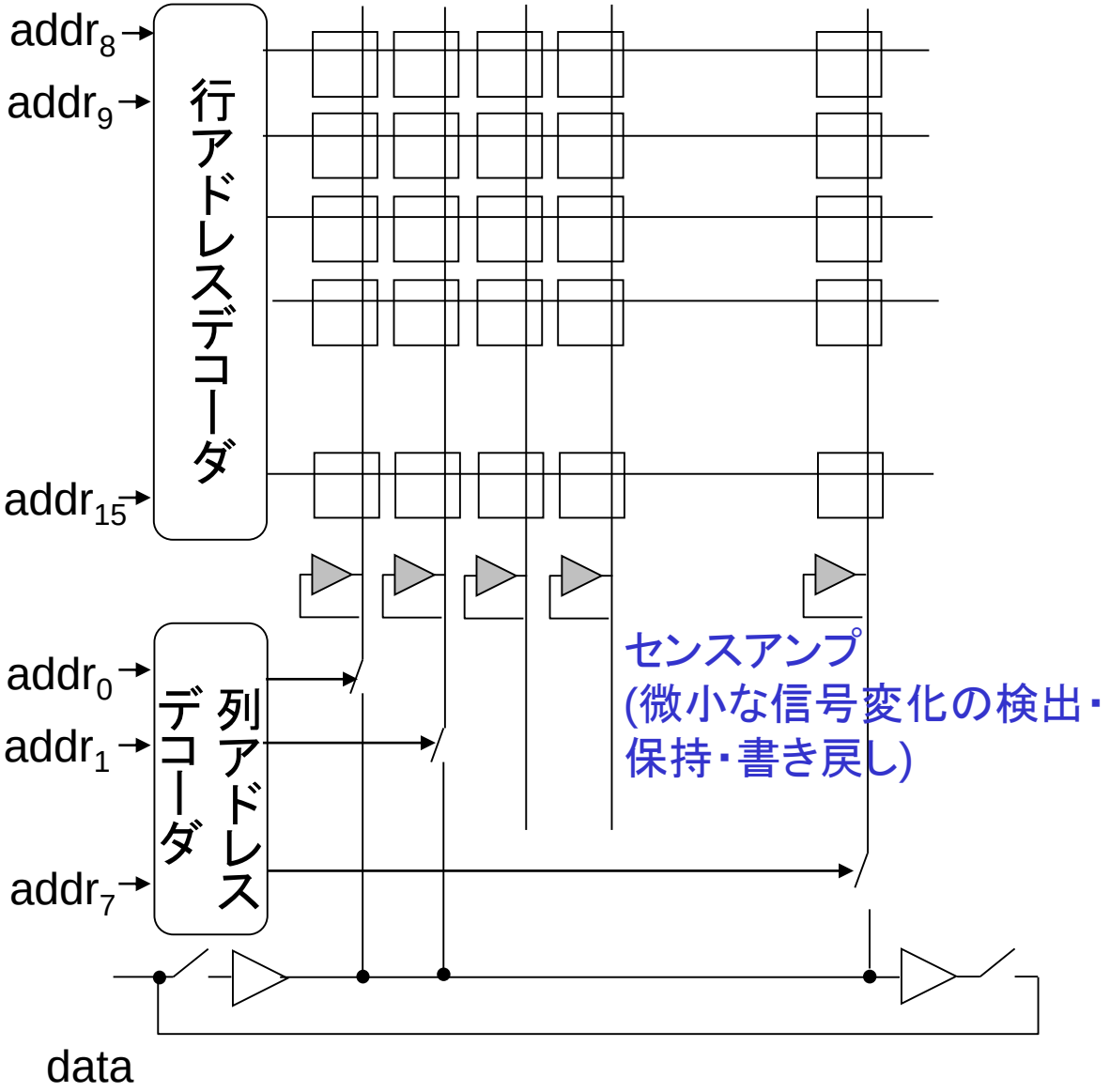
- 読み出し時には，ビット対線の電位差を増幅して値を得る
- 書き込み時には，セルのNOTゲートよりも強くビット線を駆動して記憶内容を上書きする



DRAM



- 読み出し時は，ビット線の電位の微小変化をセンスアンプが検知して増幅・保持する．選択列の値が読み出された後，センスアンプが元の値をキャパシタに書き戻す
- 書き込み時は，読み出し時と同じ動作の後，選択列のビット線のみ入力電圧で上書きする



SRAM vs DRAM

SRAM

- 1セルの回路が大きい
- 制御が比較的簡単
- 記憶内容は，電源が入っている限り安定

- よって，速いが小容量

DRAM

- 1セルの回路が小さい
- 制御が比較的複雑（信号が微弱なことと，破壊読み出しであることに起因）
- 時間が経つと記憶が消える（リフレッシュと呼ばれる再書き込み動作を数ミリ秒に1回行う必要がある）

- よって，遅いが大容量

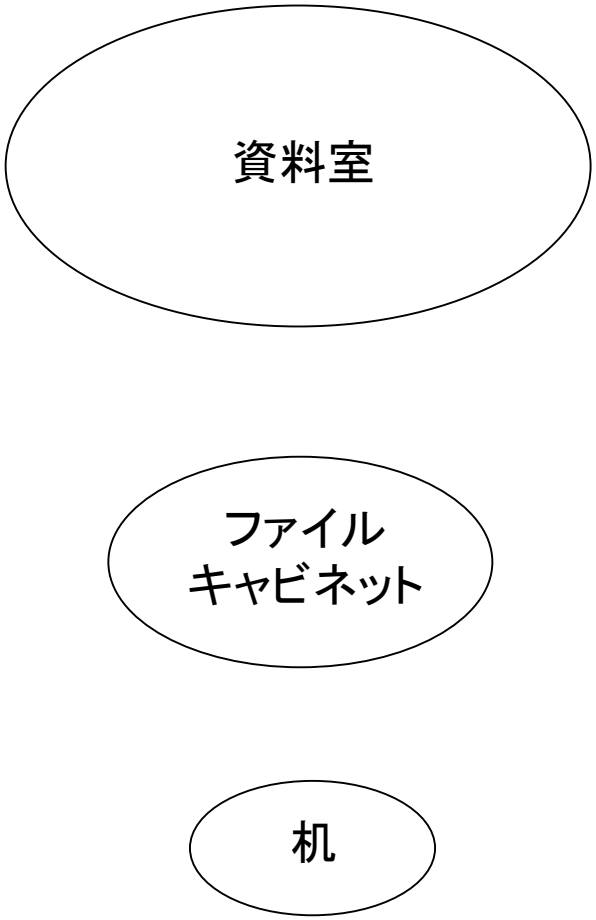
記憶階層

一般論として
- 記憶装置は小容量だと速く，大容量だと遅い．
- アクセス開始には時間がかかり，連続データのアクセスは速い．

	レイテンシ（遅延時間）	容量
ネットワーク上の記憶	~ ∞	~ ∞
ハードディスクドライブ	~ 10 ms	~ Tbytes
DRAM	~ 100 ns	~ Gbytes
SRAM	~ 10 ns (1 ~ 10クロック)	K ~ Mbytes
レジスタ	~ 1 ns (1 クロック)	32 ~ 128 bytes

よく使うものは速い記憶装置に置きたい．しかしサイズは限られている．

デスクワークからの類推

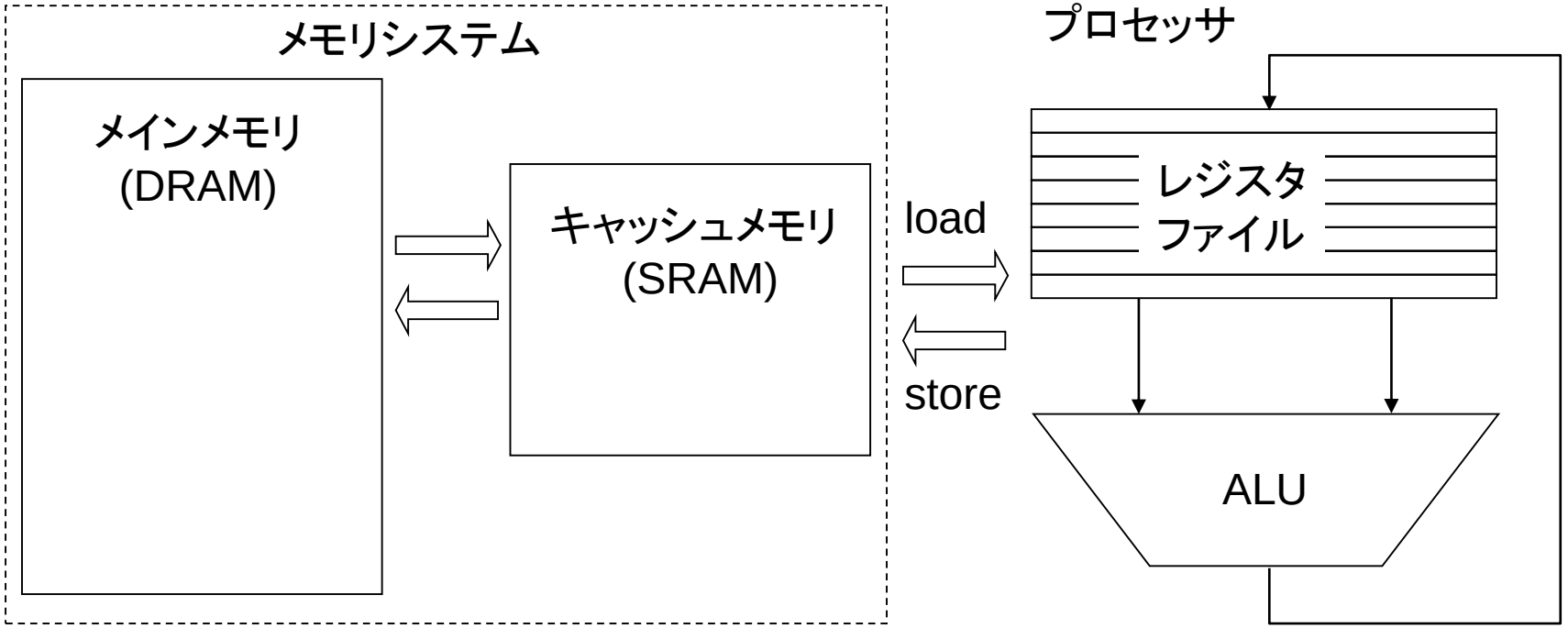


- 机のサイズは限られているので，適宜，室内のファイルキャビネットや，社内の資料室に書類を取りにいかなくてはならない
- 新しい書類が必要になったら，当面不要なものをキャビネットまたは資料室に仕舞わなくてはならない．
- さてどうするか？

自然な戦略:

- **一度使った資料はまたすぐ使う可能性が高い**ので，すぐにしまわずに机に置いておく（あるいは資料室まで戻さずにキャビネットに置いておく）
- **関連する資料がすぐ必要になる可能性が高い**ので，ある資料が必要なときには，それを綴じてあるファイルブックごと机に持ってくる

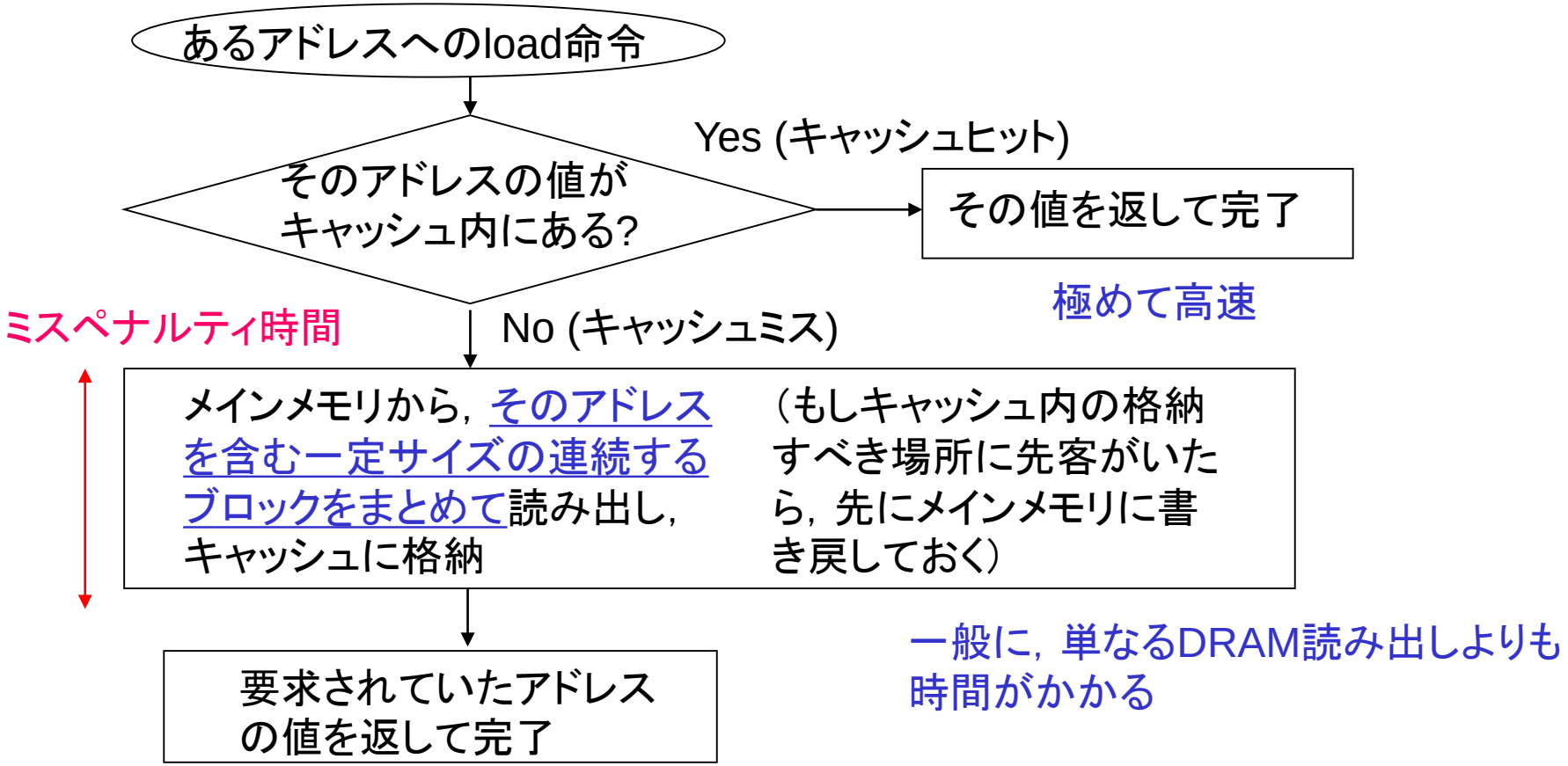
キャッシュメモリ



キャッシュメモリの制御は，以下の経験則を利用して自動的に行われる

- **時間的局所性**
  あるデータがアクセスされる場合，近いうちにその同じデータが再度アクセスされる可能性が高い
- **空間的局所性**
  あるデータがアクセスされた場合，その周囲の値もアクセスされる可能性が高い

キャッシュメモリの動作例

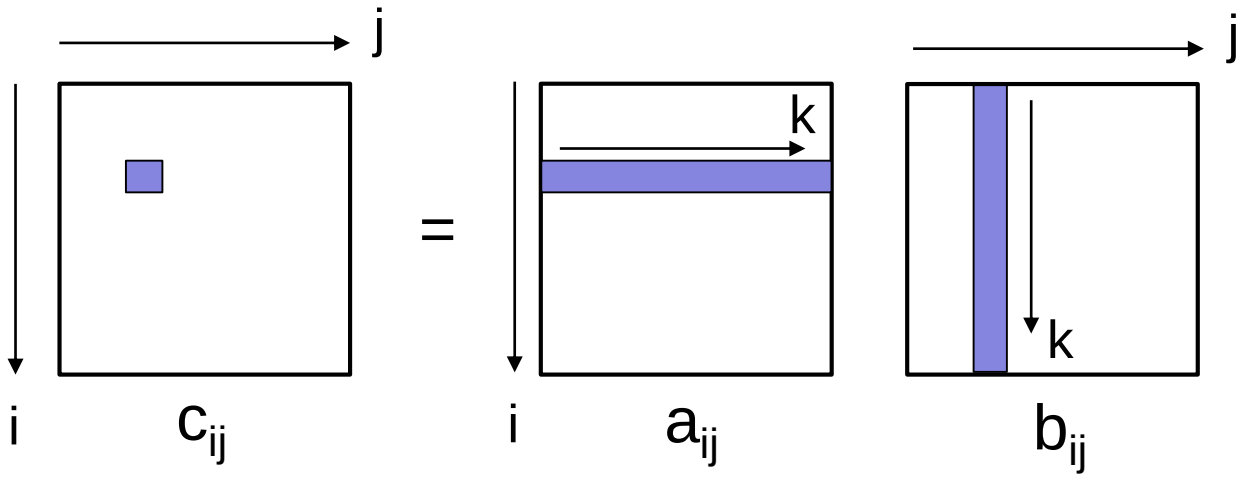


平均メモリアクセス時間
＝ ヒット時間 ＋ キャッシュミス率 × ミスペナルティ時間

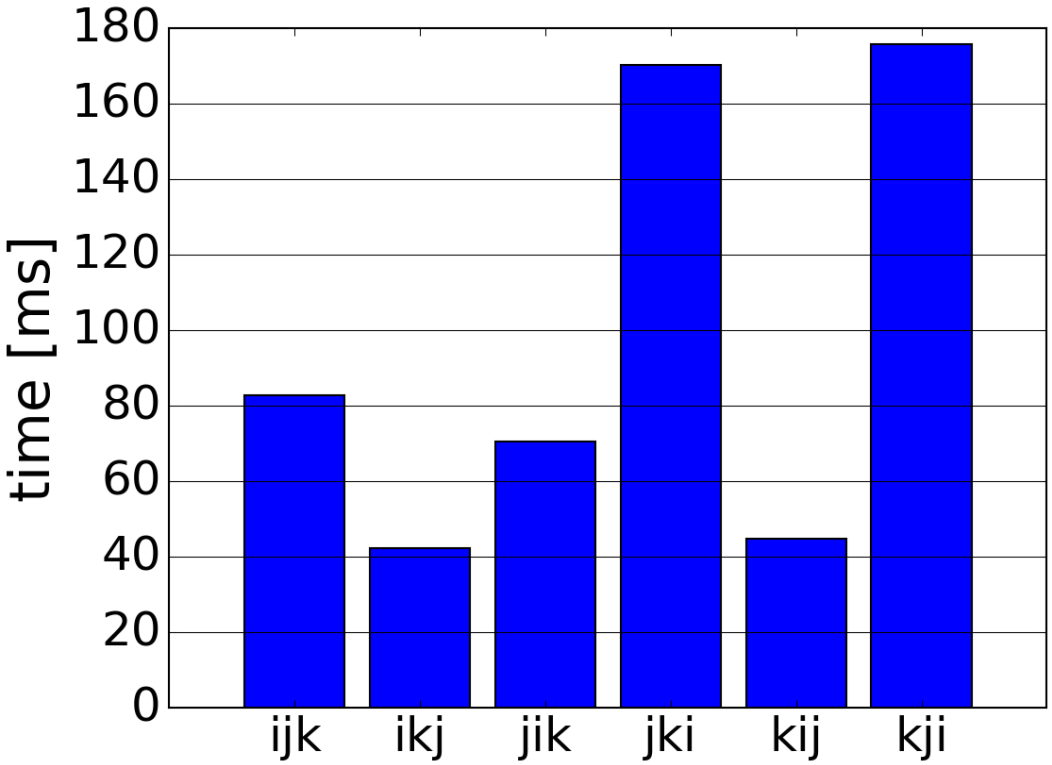
キャッシュメモリの効果: 行列積の例

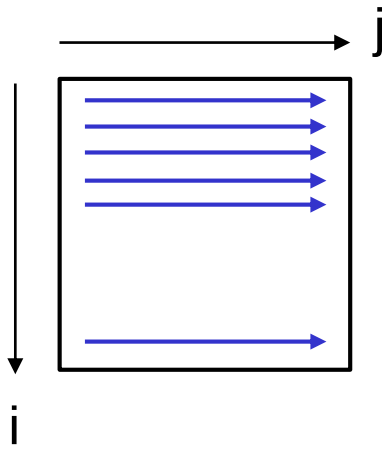
```
#define N 500
double a[N][N], b[N][N], c[N][N];

for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
        for (int k = 0; k < N; k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

この2行を入れ替える方が速く計算できる



Core i7-7600U 並列処理なしの場合





`x[i][j]` の要素の
メモリ上での並び順

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

最も内側のループで `i` や `k` が変化すると不連続なメモリアクセスが発生する

メモリの分類

ランダムアクセスメモリ vs シーケンシャルアクセスメモリ

- 任意の順序で読み書きできるものを RAM (Random Access Memory) と呼ぶ
- 最近の「メモリ」はほぼ例外なくランダムアクセス可能

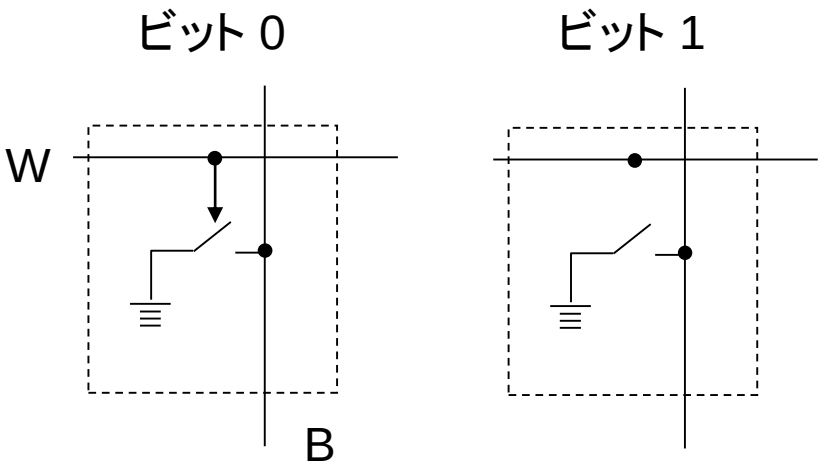
揮発性メモリ vs 不揮発性メモリ

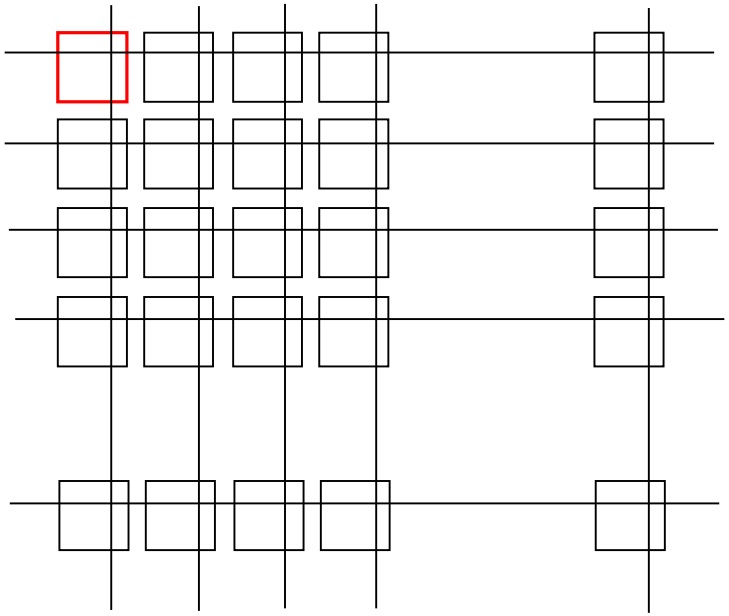
- 電源を切るとデータが消えるのが揮発性メモリ
- 不揮発性メモリのうち，主に読み出しに用いるものを ROM (Read Only Memory) と呼ぶ
  - マスクROM (半導体製造時に内容を決めてしまう)
  - PROM (Programmable ROM): 書き込み可能
  - EPROM (Erasable PROM): 消去も可能
    - UV-EPROM: 紫外線で消去
    - EEPROM: 電気的に消去（e.g. フラッシュメモリ）

※ RAM と ROM は対義語ではない（ほとんどの ROM はランダムアクセス可能）
※「メモリ」という名前でも実は「二次記憶装置」の場合がある (e.g. USBメモリ)

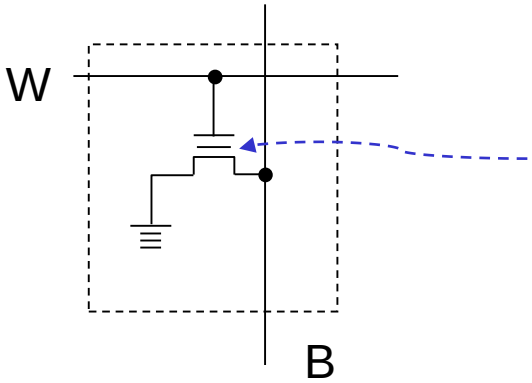
マスクROM と EEPROM

マスクROMの構成例





EEPROMの構成例



フローティングゲートと呼ばれる部分の電荷の有無によって，ゲートに電圧をかけてもスイッチオンできなくすることができる

ファミリーコンピュータ用ROMカートリッジ（ロムカセット）

http://ja.wikipedia.org/wiki/
ファイル:Famicom_ROM_cassette.jpg

http://blog.livedoor.jp/game_retro/archives/1403347.html

練習問題

1. ヒット時間が 1 ns, ミスペナルティ時間が 20 ns のメモリシステムを考える. キャッシュミス率が 5 % のときの平均メモリアクセス時間を求めよ.
2. 1のシステムにおいて, 平均メモリアクセス時間を 1.5 [ns] にするために必要なキャッシュミス率を求めよ.
3. 一般にキャッシュメモリのサイズを大きくするとキャッシュミス率は下がるが, ヒット時間は増大する傾向にある. ある計算機の設計において, キャッシュサイズを 2 倍にすることによってキャッシュミス率が 5 % から 4 % に改善することがわかった. これによって平均メモリアクセス時間を短縮できるためには, ヒット時間の増大はどの程度に抑えられている必要があるか述べよ. ただしミスペナルティ時間は変更前のヒット時間の 20 倍で, キャッシュサイズに依存しないとする.

解答例

平均メモリアクセス時間 = ヒット時間 ＋ キャッシュミス率 × ミスペナルティ時間

1. $1 + 5 \times 10^{-2} \times 20 = 2$ [ns]

2. $1 + p \times 10^{-2} \times 20 = 1.5$ を p について解いて，$p = 2.5$ [%]

3. 変更前，変更後の平均メモリアクセス時間を t_{ma1}，t_{ma2}，同じくヒット時間を t_{hit1}，t_{hit2} と書くと，

$$t_{ma1} = t_{hit1} + 5 \times 10^{-2} \times 20 \times t_{hit1}$$
$$t_{ma2} = t_{hit2} + 4 \times 10^{-2} \times 20 \times t_{hit1}$$

$$t_{ma2} - t_{ma1} = t_{hit2} - t_{hit1} - 1 \times 10^{-2} \times 20 \times t_{hit1}$$
$$= t_{hit2} - t_{hit1} \times 1.2$$

よって 1.2 倍までの増大は許容できる．(逆に言うと，ヒット時間がそれ以上増大してしまうなら，ミス率改善の努力は無駄になる)