

計算機システムの基礎(第10回 配布)

第7章2節

コンピュータの性能の推移

- (1) コンピュータの歴史
- (2) コンピュータの性能
- (3) 集積回路の進歩
- (4) アーキテクチャ

第4章 プロセッサ

- (1) プロセッサの基本機能
- (2) プロセッサの構成回路
- (3) コンピュータアーキテクチャ

第5章 メモリアーキテクチャ

担当: 福井大学 大学院工学研究科
 情報・メディア工学専攻
 森 眞一郎 (moris@u-fukui.ac.jp)

スライドは、一部のページを除き下記URLでカラー版が入手可能です
<http://sylph.fuis.u-fukui.ac.jp/~moris/lecture/ComplIntro/>

Web版では写真カット

1. コンピュータの世代

計算する機械

電子式

解析機関by Babbage 手回し式計算機
 @19世紀 未完成 @1903 日本製!

ABC@1939
 (真空管でALU単機能)

ENIAC@1946
 [1800本, 150KW, 30t]
 (真空管式 プログラム可)

10桁の
 加算 5400回/s
 乗算 300回/s

von Neumann方式

EDSAC@1939 [3000本, 12kW, 20m²]
 (真空管でプログラム内蔵方式)

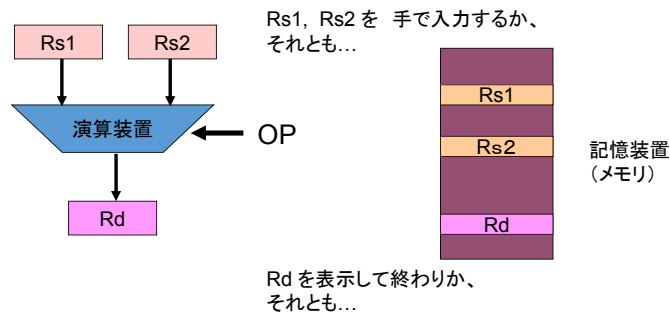
ETL MarkIII@1956
 (世界初のTr式。日本製)

Intel 4004@1971 [108KHz]
 (4bitマイクロプロセッサ)
 日本のビジコン と Intelが共同開発
 [設計者は 嶋正利]

電卓 から コンピュータへ

基本は2項演算

Rd ← Rs1 OP Rs2



電卓 から コンピュータへ

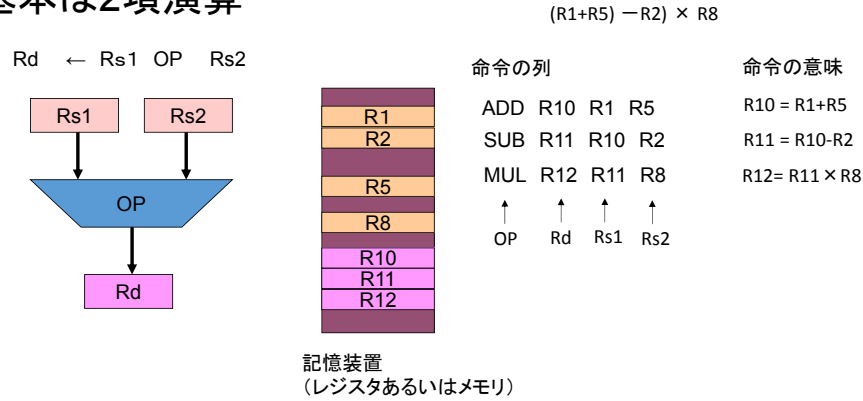
コンピュータの命令の例

命令コード	オペランド1	オペランド2	オペランド3
-------	--------	--------	--------

- 命令コード(OP Code: Operation Code)
命令の種類を表す
- N個のオペランド(Operand) (通常 N = 3 or 2 [or 1 or 0])
命令が使用するデータを指定する
(レジスタや主記憶に格納された変数や定数)

電卓 から コンピュータへ

基本は2項演算



2+3. 集積回路の進歩とコンピュータの世代

• 半導体の微細化技術

• 「ムーアの法則」

...Gordon Moore@Intelが発表した半導体技術に関する経験則

⇒ 半導体の集積度は Y 年で 2倍になる。

かつて

- DRAMの容量は1.5年で2倍 (3年で4倍) $Y=1.5$
- マイクロプロセッサのトランジスタ数は 2年で2倍 $Y=2$



最近では、微細化技術が多くの難題に直面し..... $Y \Rightarrow 3$

Web版ではデータカット

メモリ容量: 10年で100倍
(ムーアの法則: 3年で4倍)
周波数: 10年で13倍

2+3. 集積回路の進歩とコンピュータの世代

Web版では写真カット

• 半導体の微細化技術

プロセッサ名	年	動作周波数	トランジスタ数
4004	1971	108K	2,300
80286	1982	12M	134,000
Pentium	1992	66M	3,100,000
Pentium IV	2000	1.5G	42,000,000
Pentium D	2005	3.4G	230,000,000
Core2 Duo	2006	2.66G	291,000,000
Core i7 2600	2011	3.4G	995,000,000
Core i7 5960X	2014	3.5G	26億

大容量
キャッシュ
の影響大

2core/1M
2core/4M
4core/(8+1)M

8core/20M

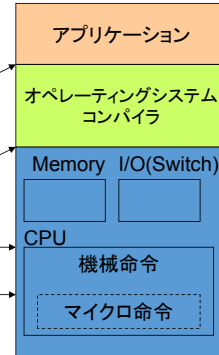
コンピュータアーキテクチャとは？ (Computer Architecture)

語源：
IBM System/360 の開発者であるG.M.Amdahl の定義

プログラマから見えるコンピュータの論理仕様

• 「見る」レベルを変えると色々なアーキテクチャが定義できる

- OS(レベル)アーキテクチャ
- PMS (Processor-Memory-Switch) (レベル)アーキテクチャ
- 命令セット(レベル)アーキテクチャ
- マイクロアーキテクチャ

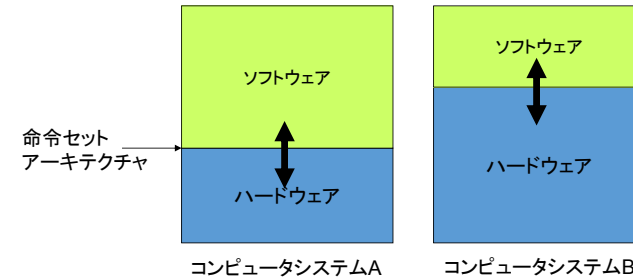


コンピュータアーキテクチャとは？ (Computer Architecture)

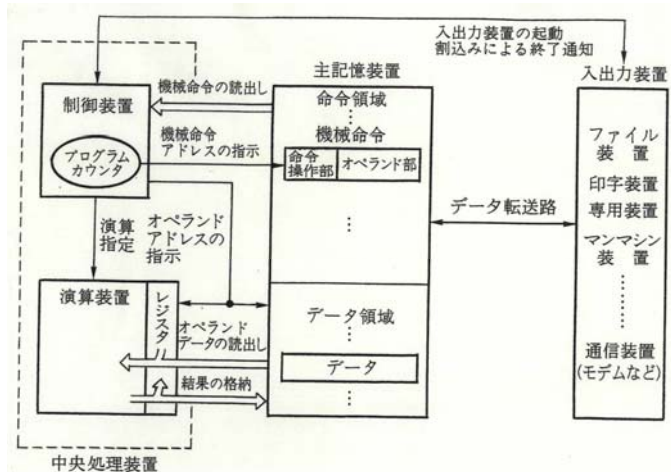
プログラマから見えるコンピュータの論理仕様

コンピュータシステムにおける
ハードウェアとソフトウェア
のトレードオフ(適切なバランスの決定)

コンピュータ
アーキテクチャ設計



コンピュータの基本構造

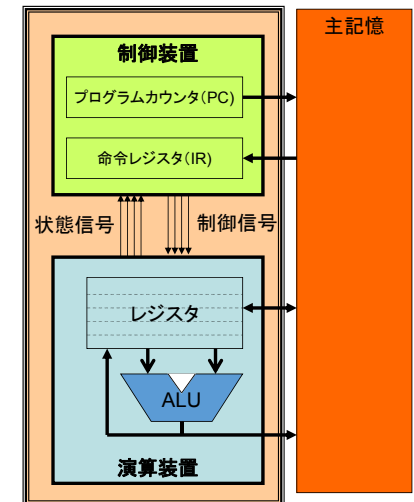


コンピュータの基本構成要素

プロセッサの基本構成と動作原理

• CPUの基本構造

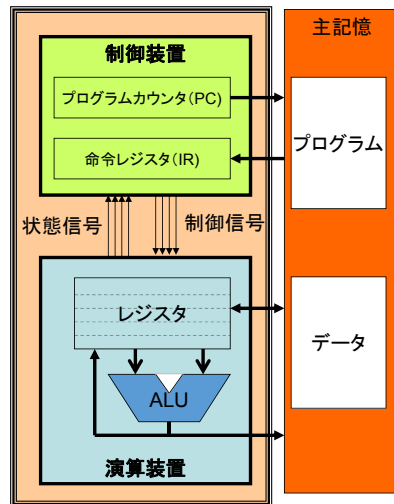
- 制御装置
 - プログラムカウンタ(PC)
 - 命令レジスタ(IR)
- 演算装置
 - 算術論理演算器(ALU)
 - 四則演算器、論理演算器
 - レジスタ



プロセッサの基本構成と動作原理

- CPUの命令サイクル
 - 命令フェッチ(IF)
 - 命令デコード(D)
 - オペランドフェッチ(OF)
 - 演算実行(EX)
 - 結果格納(S)

これらの幾つかを1つのサイクルにまとめることもある
 例) DをIFに含める
 OFをDに含める
 SをEXに含める

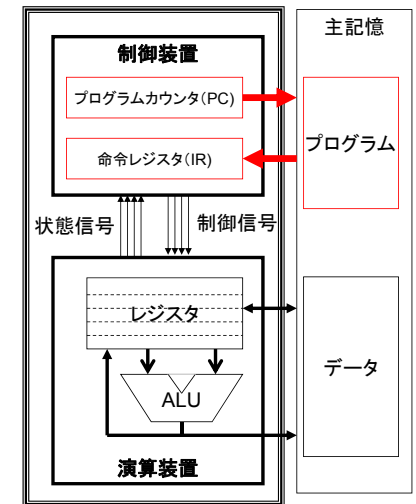


プロセッサの基本構成と動作原理

- CPUの命令サイクル
 - **命令フェッチ(IF)**
 - 命令デコード(D)
 - オペランドフェッチ(OF)
 - 演算実行(EX)
 - 結果格納(S)

動作

PCが指し示す主記憶上のアドレス(番地)から命令を読みだして、命令レジスタに格納

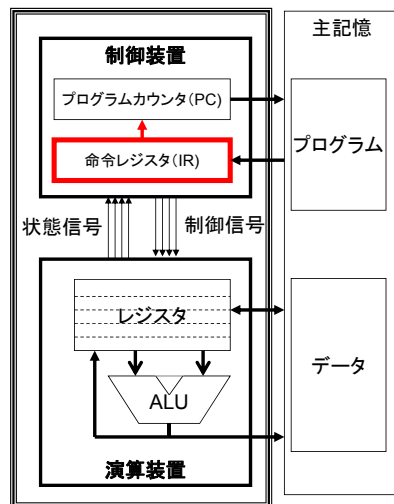


プロセッサの基本構成と動作原理

- CPUの命令サイクル
 - 命令フェッチ(IF)
 - **命令デコード(D)**
 - オペランドフェッチ(OF)
 - 演算実行(EX)
 - 結果格納(S)

動作

命令レジスタに格納された命令を解析し、
 ・実行すべき命令の種類、
 ・オペランドが格納されている
 主記憶あるいはレジスタのアドレス(番地)を得る

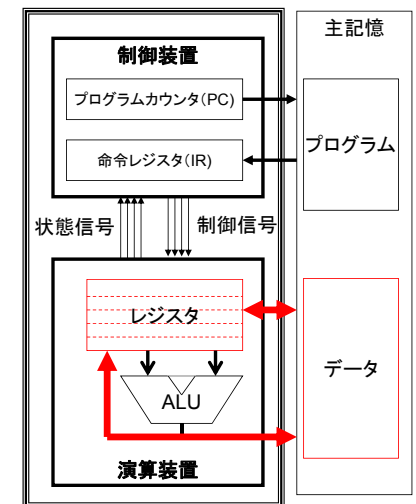


プロセッサの基本構成と動作原理

- CPUの命令サイクル
 - 命令フェッチ(IF)
 - 命令デコード(D)
 - **オペランドフェッチ(OF)**
 - 演算実行(EX)
 - 結果格納(S)

動作

計算に必要な入力オペランドを
 主記憶あるいはレジスタ上のアドレス(番地)
 から読みだし、演算器(ALU)へ送る

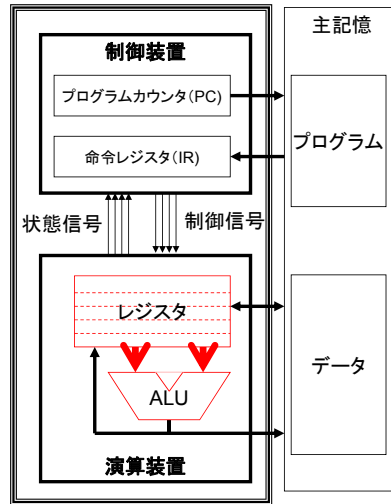


プロセッサの基本構成と動作原理

- CPUの命令サイクル
 - 命令フェッチ(IF)
 - 命令デコード(D)
 - オペランドフェッチ(OF)
 - **演算実行(EX)**
 - 結果格納(S)

動作

与えられた入力オペランドに対して、現在の命令に対応する「演算」を行う。

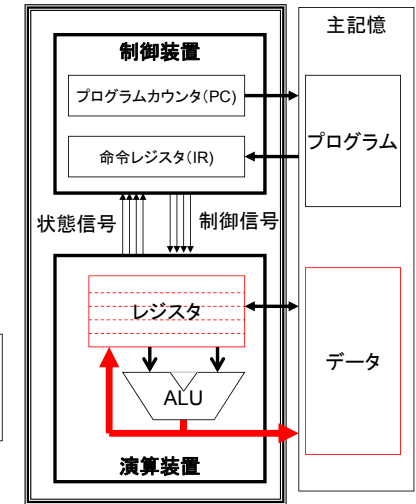


プロセッサの基本構成と動作原理

- CPUの命令サイクル
 - 命令フェッチ(IF)
 - 命令デコード(D)
 - オペランドフェッチ(OF)
 - 演算実行(EX)
 - **結果格納(S)**

動作

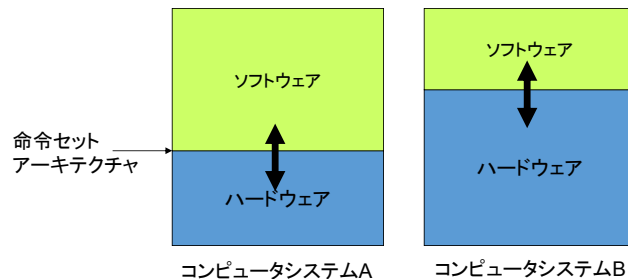
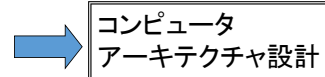
演算結果を主記憶あるいはレジスタ上のアドレス(番地)に格納
(主記憶からの読み出し命令の場合は、主記憶から読みだしたデータを、レジスタに格納)



[復習]コンピュータアーキテクチャとは? (Computer Architecture)

プログラマから見えるコンピュータの論理仕様

コンピュータシステムにおける
ハードウェアとソフトウェア
のトレードオフ(適切なバランスの決定)



命令セットアーキテクチャ

教科書
4.1節に
相当

命令セットが備えるべき基本的な機能

命令コード(OP)の種類と考えてOK

- 演算命令
 - 四則演算、論理演算、シフト演算、など
 - 例えばComet IIでは
ADD*, SUB*, AND, OR, XOR, SL*, SR* 等
- データ転送命令
 - 主記憶からレジスタへの転送命令「LOAD命令」 LD
 - レジスタから主記憶への転送命令「STORE命令」 ST
- プログラム制御命令
 - 特定のアドレスの命令に分岐する「分岐命令」 JPL, JNZ, JUMP等
 - サブルーチン(関数)を呼び出す「手続き呼び出し命令」 CALL
 - サブルーチンから復帰する「リターン命令」 RET
 - など
- その他の命令
 - 入出力命令、OS呼び出しなどのシステム制御用命令 SVC

メモリアーキテクチャ

- メモリデバイスの性質
 - 「アクセス速度」と「記憶容量」のトレードオフ
 - 高速なメモリは、容量を大きくできない。
 - 大容量のメモリは、高速動作できない。
- メモリアクセスの性質
 - 「参照の(時間・空間)局所性」
 - あるデータが一度アクセス(読み出し・書き込み)されると
 - そのデータの、**近くのデータ**がアクセスされる確率が高い
 - (空間局所性)
 - そのデータを、**近い将来**再びアクセスする確率が高い
 - (時間局所性)

例えば
for (k=0;k<N){
 A[i] = A[i] + B[i][k]*C[k];
}

21

メモリアーキテクチャ

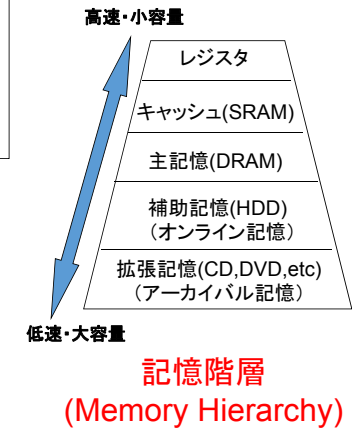
メモリデバイスの性質
・「アクセス速度」と「記憶容量」のトレードオフ

メモリアクセスの性質
・「参照の(時間・空間)局所性」



メモリアーキテクチャの基本方針

これらの性質を利用して
CPUの近くには高速・小容量のメモリを
CPUの遠くには低速・大容量のメモリを
物理的に配置し、アクセス頻度に応じて
データのおき場所を再配置する。



22