

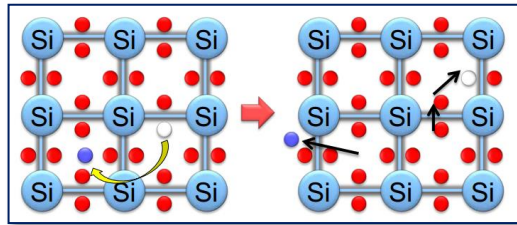


コンピュータシステム (アーキテクチャ 第3回)



工学部 情報エレクトロニクス学科
大学院 情報科学研究所 情報理工学部門
堀山 貴史

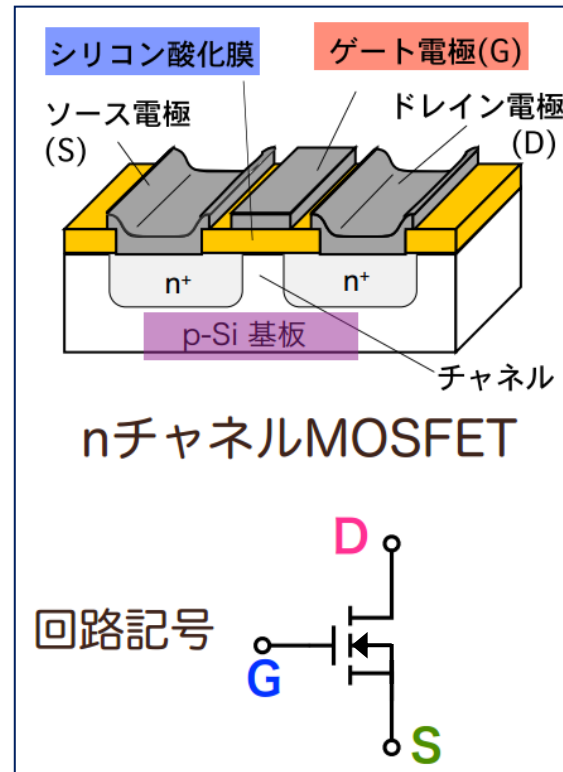
(参考) 半導体から論理ゲートへ



半導体(固体物理)
(有田先生の講義より)

- 導体
電気が流れやすい
- 絶縁体
電気がほとんど流れない
- 半導体(その中間)
条件により電気が流れる

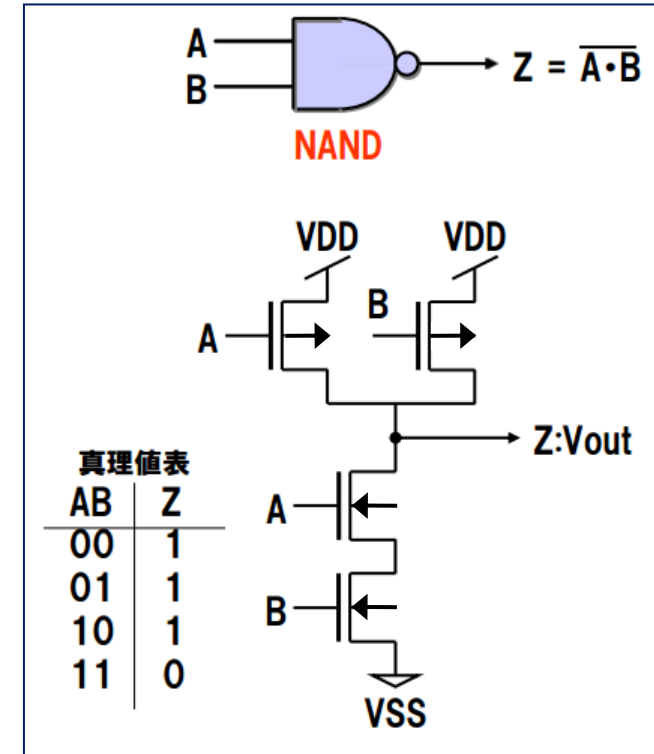
アーキテクチャ #3



トランジスタ
(電子デバイス技術)
(本久先生の講義より)

電流を制御

コンピュータシステム



論理ゲート
(トランジスタ回路技術)
(池辺先生の講義より)

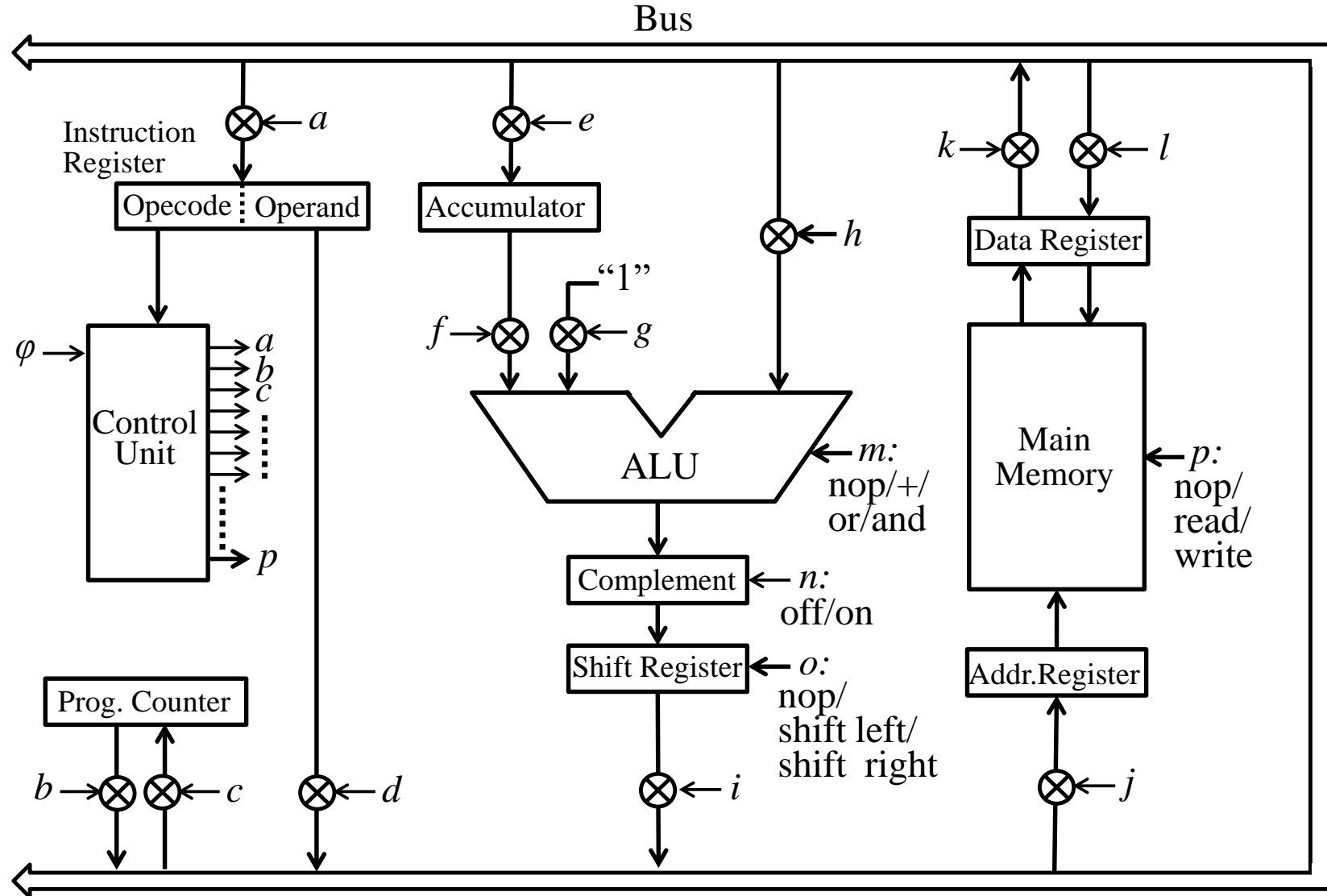
論理演算を実現

前回(アーキテクチャ第2回)の内容

機械語命令と内部動作

- 基本的・典型的なアーキテクチャの構成
- 一般的な機械語命令の形式
- 機械語命令に対するレジスタ転送レベルの内部動作
 - フェッチ動作
 - ロード命令
 - 加算命令
 - 減算命令(→ 補数表現)
 - ストア命令
 - 条件分岐命令
 - 停止命令と割り込み動作

基本的・典型的なアーキテクチャの構成



今回の内容

機械語命令と内部動作(2)

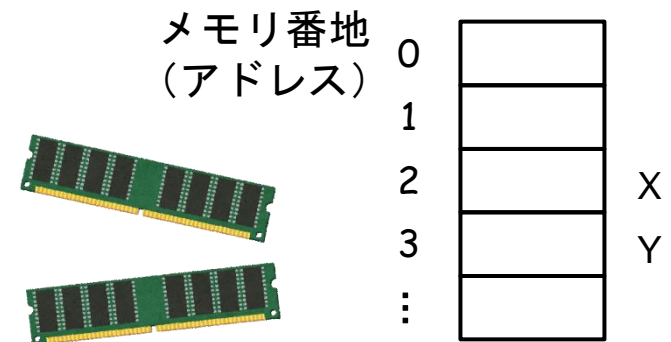
- 主記憶アドレス参照方式

アーキテクチャの基本知識(1) (分類と概観・初期のメインフレーム)

- 価格・時代・用途による汎用計算機の分類:
 - メインフレーム、ミニコン・ワークステーション、パソコン
- 黎明期～初期のメインフレーム技術
 - 機械式から電子式へ
 - ハードワイアドとノイマン型計算機
 - バッチ処理とタイムシェアリング
 - IOプロセッサとバス構成
 - ファミリー思想

コンピュータ内部で、どう実現する？

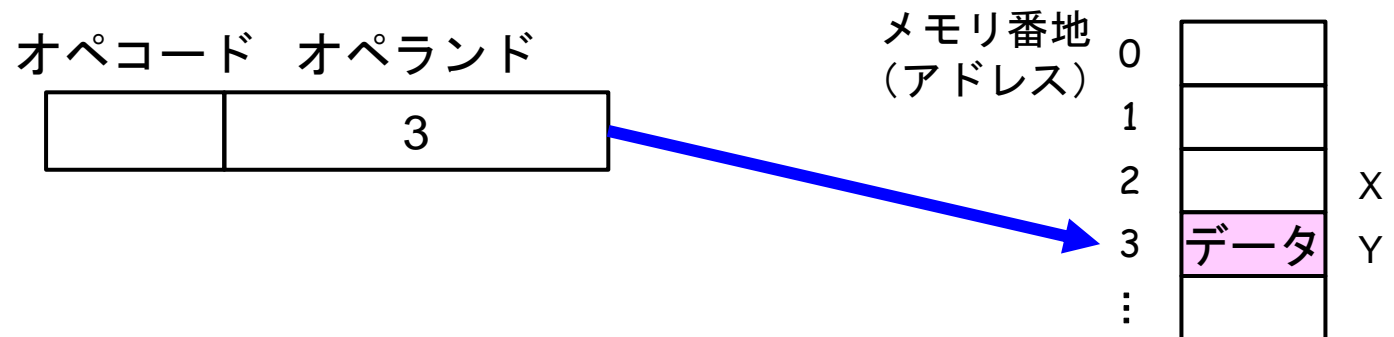
- $X = X + Y$
 - 変数の値は、できればレジスタで保持したい
(レジスタは、主記憶(メモリ)よりずっと高速)
 - でも、レジスタ数は有限
→ 主記憶に保持して、レジスタをやりくり
- 主記憶 (メモリ) の番地 (**アドレス**) を**指定**してその値を利用する



主記憶アドレス参照方式

- 直接アドレス方式

- 機械語の一部(オペランド)に書かれた番地のデータを読み出して使う
(ハードウェアは簡単になるが、使うときの柔軟性に欠ける)



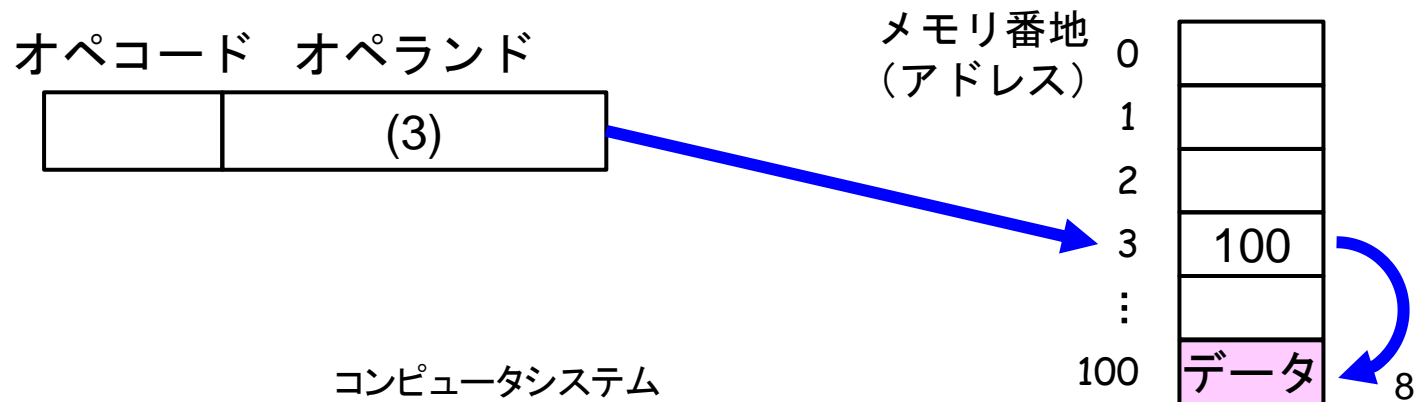
主記憶アドレス参照方式

- 直接アドレス方式

- 機械語の一部(オペランド)に書かれた番地のデータを読み出して使う
(ハードウェアは簡単になるが、使うときの柔軟性に欠ける)

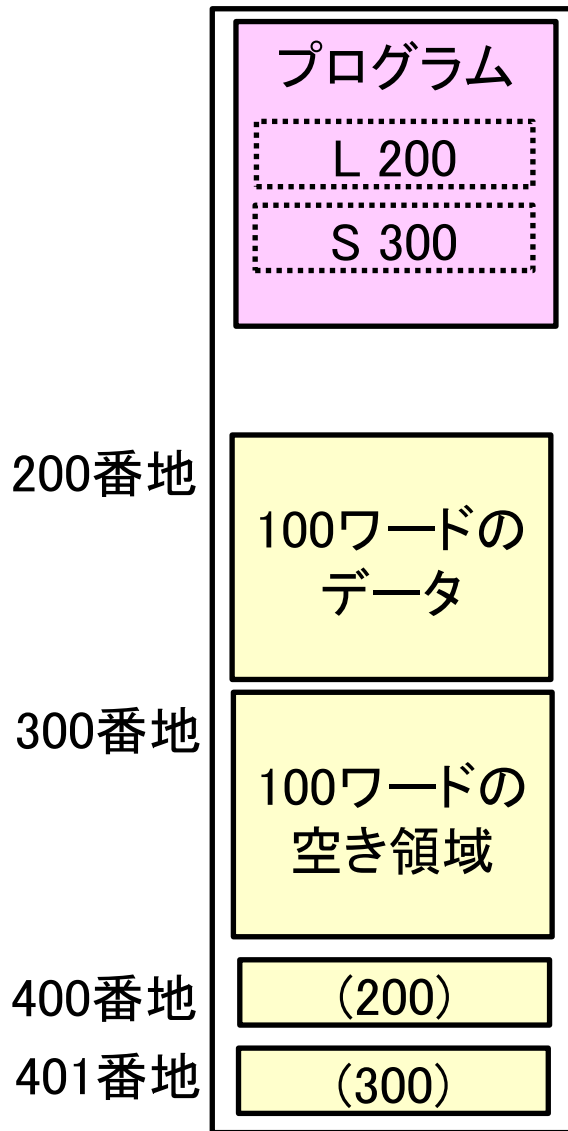
- 間接アドレス方式

- オペランドに書かれた番地のデータ内容を読み出し、それをメモリの番地として再度参照し、そこに格納されたデータを読み出して使う

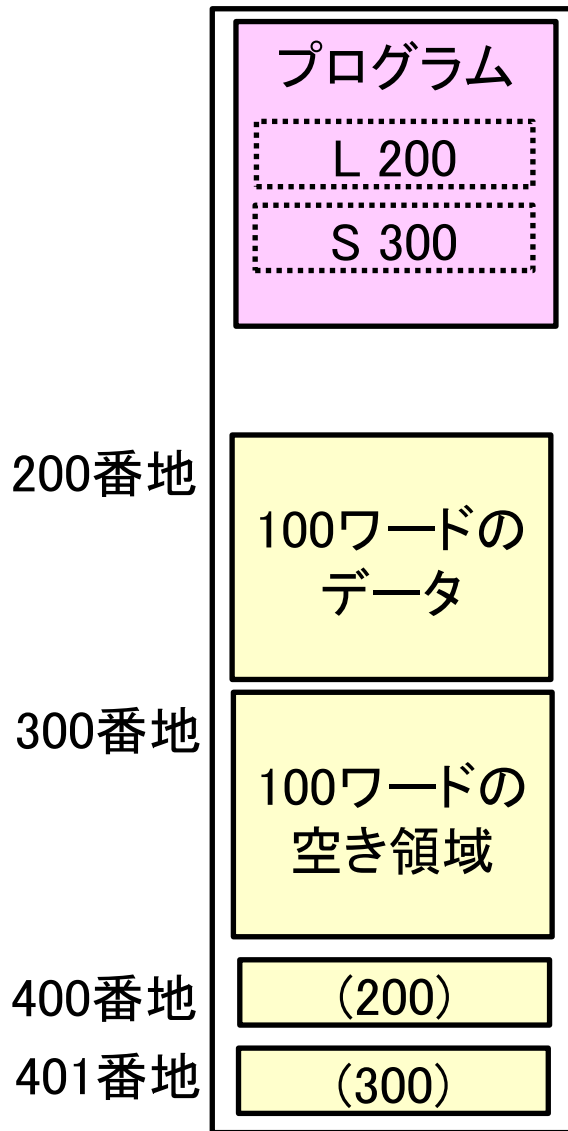


間接/インデクスアドレス方式が欲しい例

- $(200+i)$ 番地のデータを $(300+i)$ 番地にそれぞれコピーしたい(ブロック転送)

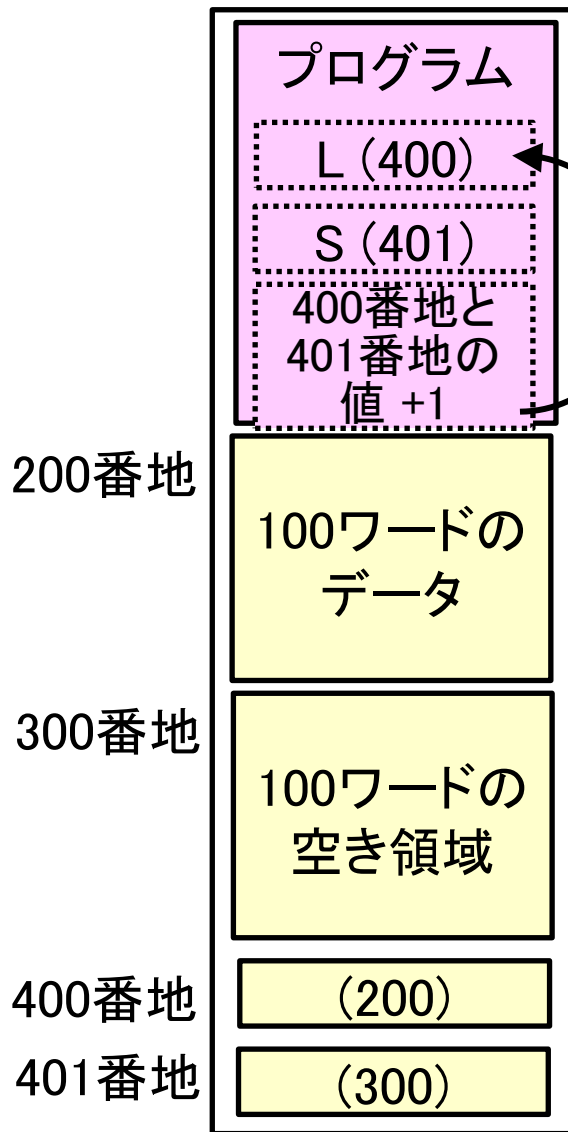


間接/インデクスアドレス方式が欲しい例



- $(200+i)$ 番地のデータを $(300+i)$ 番地にそれぞれコピーしたい (ブロック転送)
 - ロード命令とストア命令を100回繰り返せば実行できる。
 - 直接アドレス方式の場合、機械語のオペランドの番地指定が毎回異なるので、単純な繰り返しプログラムでは書けない。
 - 自分自身の機械語を書き換えるプログラムを作れば実行できるが、ソフトウェアの維持管理が面倒になる。

間接/インデクスアドレス方式が欲しい例



- $(200+i)$ 番地のデータを $(300+i)$ 番地にそれぞれコピーしたい (ブロック転送)
 - ロード命令とストア命令を100回繰り返せば実行できる。
 - 直接アドレス方式の場合、機械語のオペランドの番地指定が毎回異なるので、単純な繰り返しプログラムでは書けない。
 - 自分自身の機械語を書き換えるプログラムを作れば実行できるが、ソフトウェアの維持管理が面倒になる。
 - 間接アドレス方式やインデクス方式が可能であれば、プログラムそのものは書き換えずにくり返し処理ができる。
- プログラムとデータの分離

主記憶アドレス参照方式

・ インデクスアドレス方式

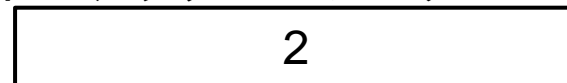
- インデクスレジスタという特別なレジスタを持ち、オペランドに書かれた番地にインデクスレジスタの数値を加えたものをメモリの番地として参照し、そこに格納されたデータを読み出して使う
- 例：配列へのアクセスに利用

A[0] 100番地
A[1] 101番地
A[2] 102番地
A[3] 103番地
⋮

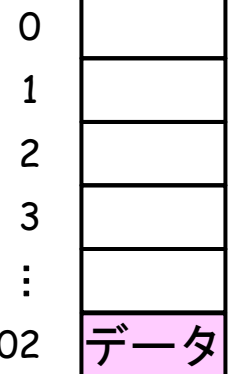
オペコード オペランド



インデックスレジスタ



メモリ番地
(アドレス)



休憩

- ここで、少し休憩しましょう。
- 深呼吸したり、肩の力を抜いてから、次のビデオに進んでください。

今回の内容

機械語命令と内部動作(2)

- 主記憶アドレス参照方式

アーキテクチャの基本知識(1) (分類と概観・初期のメインフレーム)

- 価格・時代・用途による汎用計算機の分類:
 - メインフレーム、ミニコン・ワークステーション、パソコン
- 黎明期～初期のメインフレーム技術
 - 機械式から電子式へ
 - ハードワイアドとノイマン型計算機
 - バッチ処理とタイムシェアリング
 - IOプロセッサとバス構成
 - ファミリー思想

価格・時代・用途による汎用計算機の分類

- ・ メインフレーム (mainframe; 主流の系列)
 - 国家や大企業の基幹業務を処理するための大型高性能計算機
 - **特注品**で千万～数億円。1フロアや建物を占有する規模。IBMなどの大企業が設計製造。1950年代～最近まで計算機技術をリード。
- ・ ミニコン/ワークステーション (mini computer/workstation)
 - 大学の研究室や小規模オフィスの業務を処理できる中型計算機
 - **市販品**で数十～数百万円。自室に置ける規模。1990年代以降、高性能化とともにメインフレーム機からの置き換えが進む。
(→メインフレーム機はもっと凄いスーパーコンピュータへ特化)
- ・ パソコン (personal computer; PC)
 - 個人用の計算機。税務処理や文書作成など。**量産品**で数万～数十万円。デスクトップ→ラップトップ→ノート→タブレット→スマホ。1980年頃から出現したが、2000年以降、携帯電話の発展と一体化。

汎用計算機アーキテクチャの歴史

- ・ 黎明期～初期のメインフレーム(1970年頃まで)
 - 機械式→真空管式→トランジスタ式
 - ノイマン型
 - バッチ処理とタイムシェアリング
- ・ メインフレームの発展(1970年～80年代)
 - マイクロプログラム・エミュレーション
 - 仮想メモリ・仮想マシン・メインフレーム互換機
 - パイプライン処理・ベクトル並列化
- ・ マイクロプロセッサ～最近(1980年代以降)
 - 集積回路技術・電卓・x86プロセッサ・インテル・Windows
 - ワークステーション・UNIX/Linux・PC互換機
 - Apple・インターネットとPC・タブレットPC・Android

今回から3回に分けて
歴史を振り返りつつ、
技術の進化のための
アイデアを紹介します

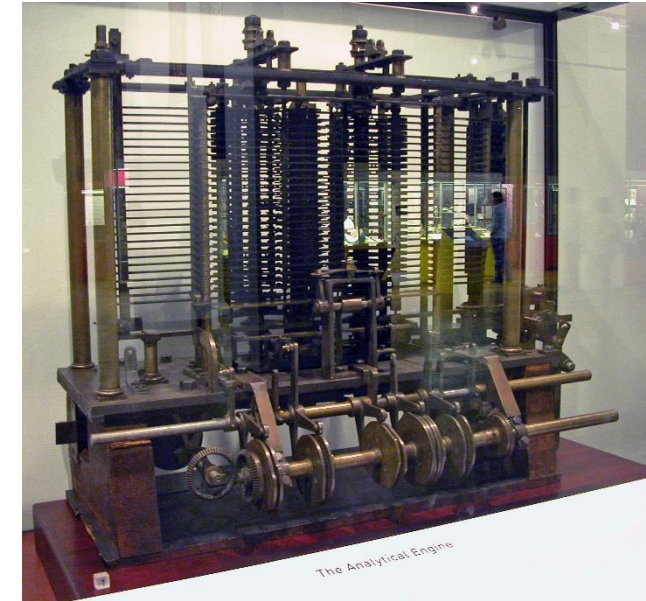
汎用計算機アーキテクチャの歴史

- ・ 黎明期～初期のメインフレーム(1970年頃まで)
 - 機械式→真空管式→トランジスタ式
 - ノイマン型
 - バッチ処理とタイムシェアリング
- ・ メインフレームの発展(1970年～80年代)
 - マイクロプログラム・エミュレーション
 - 仮想メモリ・仮想マシン・メインフレーム互換機
 - パイプライン処理・ベクトル並列化
- ・ マイクロプロセッサ～最近(1980年代以降)
 - 集積回路技術・電卓・x86プロセッサ・インテル・Windows
 - ワークステーション・UNIX/Linux・PC互換機
 - Apple・インターネットとPC・タブレットPC・Android

今回から3回に分けて
歴史を振り返りつつ、
技術の進化のための
アイデアを紹介します

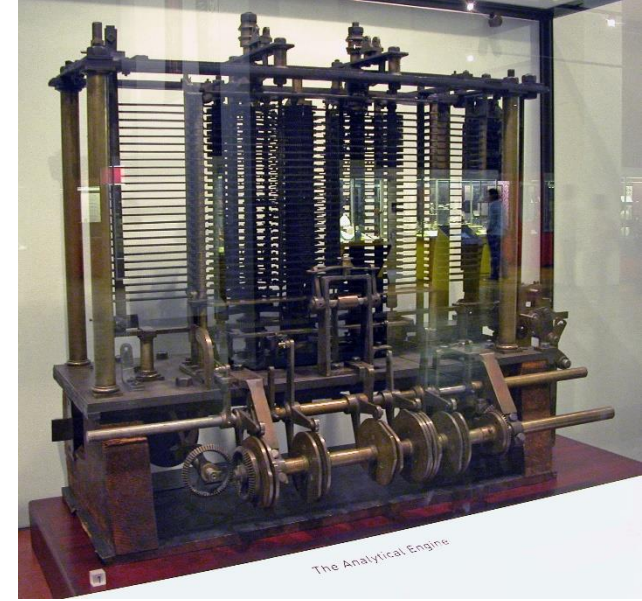
機械式から電子式へ

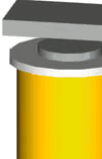
- ・ 機械(歯車)式計算機 (1600年代:江戸時代～1800年代)
 - パスカル(Pascal)の卓上計算機、
ライプニッツ(Leibniz)の歯車
 - オルゴール、自動人形(オートマトン)、
からくり人形(江戸時代)
 - バベッジ(Babbage)の階差機関
(19世紀・蒸気エンジンを利用)
(パンチカードによるソフトウェア記述)

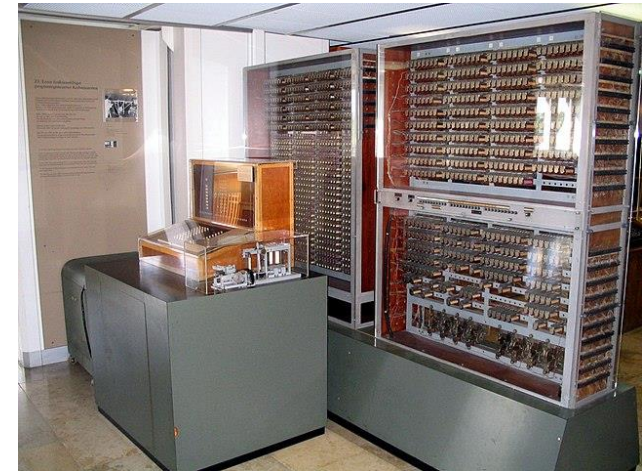
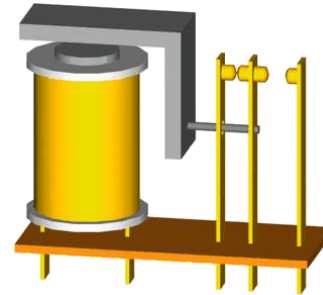


機械式から電子式へ

- ・ 機械(歯車)式計算機(1600年代:江戸時代~1800年代)
 - パスカル(Pascal)の卓上計算機、ライプニッツ(Leibniz)の歯車
 - オルゴール、自動人形(オートマトン)、からくり人形(江戸時代)
 - バベッジ(Babbage)の階差機関(19世紀・蒸気エンジンを利用)(パンチカードによるソフトウェア記述)



- ・リレー式計算機（1940年頃）
 - － 電磁石スイッチで論理関数を表現。
歯車のような摩擦はないが、
機械的接点を持つため、
演算速度には限界
- 



- ## ・電子式計算機

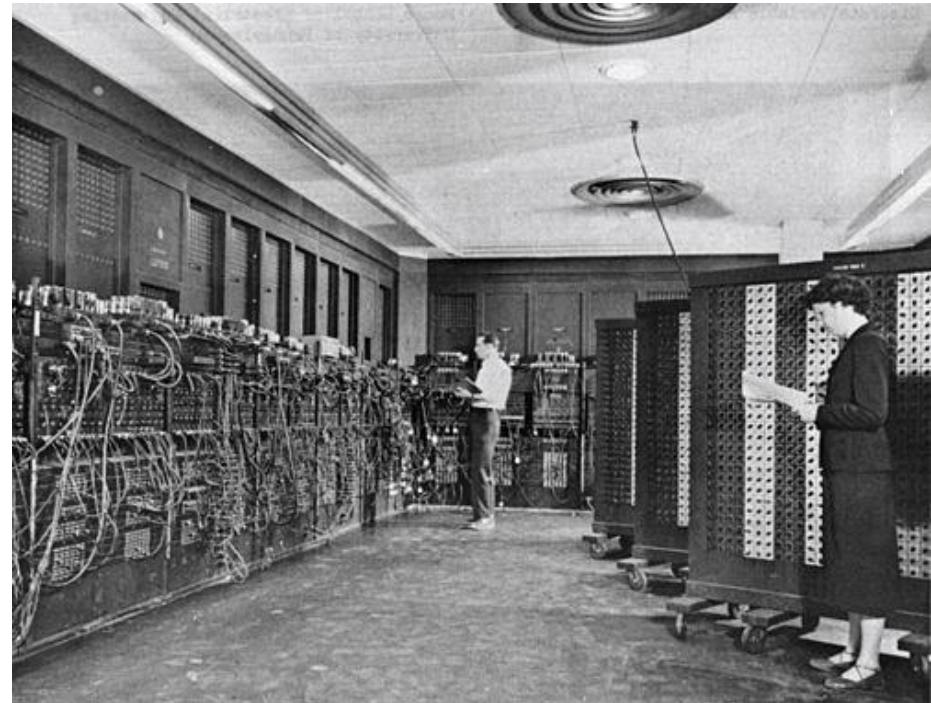
機械式から電子式へ

- ENIAC (米国ペンシルベニア大1943～1946) 第2次世界大戦中
 - 最初の汎用電子計算機の1つ
 - 重量30t, 床面積200m³, 真空管18000本, 消費電力140kW
(真空管の寿命は1万時間くらい。維持管理コストは膨大)
 - プログラムの入力はスイッチ設定やケーブル抜き差しが必要
 - 主目的: 第二次大戦の大砲の弾道計算表の作成

(参考) 第2次世界大戦の前後に
電子計算機が登場した

- ACE, Mark I (1946年, 英国)
 - Alan Turing が参加
- EDSAC (1949年, 英国)
 - von Neumann が影響
 - Wilkes ら (1967年 Turing賞)

Computer science 分野の
ノーベル賞のように思ってください



(脱線) チューリング博士が考えたこと(1)

コンピュータ
サイエンス
のはじまり



(Alan M. Turing, 1912-1954, 英国)

- 情報科学の開拓者
- チューリング機械と
計算可能性理論の創始
(1930年代)
- 米英の暗号解読計画
(**エニグマ計画**)に従事
(1940年代)
- 初期の**デジタル計算機**
開発に従事(英国ACE,
Mark I)
- チューリングテスト,
自己増殖機械,
機械学習 などの
創始者としても有名

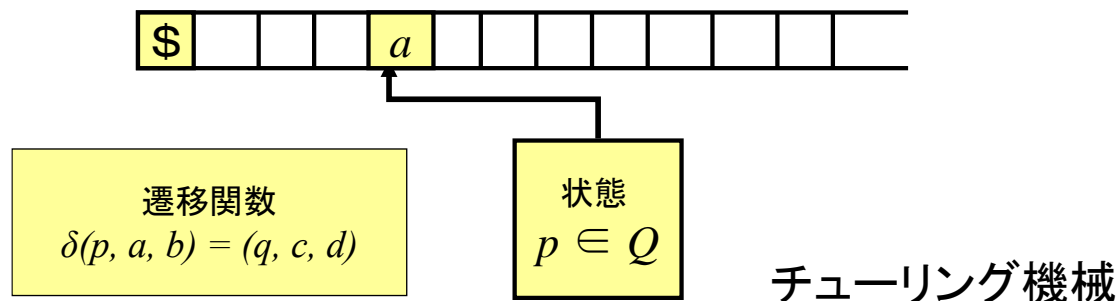
(脱線) チューリング博士が考えたこと (2)

コンピュータ
サイエンス
のはじまり

- 「考える」 = 「計算する」と考えた
手で計算する人 (computer) の動作を考えて、
チューリング機械を発明
- 何が計算できるか／できないかの理論を創始
- 万能チューリング機械は、のちの**プログラム内蔵式
計算機**(電子計算機)を予見。天才たちが続々と続いた



(Alan M. Turing, 1912-1954,
英国) Wikipedia より



ハードワイアドからノイマン型計算機へ

- ・ ENIACは、電子回路で計算手順を表現していた
 - **ハードワイアド(hard wired)**方式の計算機
(データはメモリに置き、計算手順は電子回路で制御)
- ・ **計算手順もデータと同様にメモリに記憶**すれば
プログラムの変更が容易になる
 - ENIACプロジェクトメンバだったノイマン(J. von Neumann)が
次期計算機EDVACの設計の際に提案(1945年)
 - **プログラム内蔵(stored program)**方式と呼ばれる
(**ノイマン型計算機**と呼ばれることの方が多い)
 - EDVACが完成したのは1951年。その前の1949年に、英国
ケンブリッジ大のEDSACが先にプログラム内蔵方式で完成
- ・ 当時の計算機設計は国家的プロジェクト(予算・期間・人材)
 - 弾道計算、暗号解読、宇宙開発、核兵器解析、その他、国の存亡に関わる問題



John von Neumann
(1903 - 1957,
ハンガリー／米国)
Wikipedia より

計算機の素子技術の発展

- ・ スイッチ素子
 - 歯車 → リレー → 真空管 → トランジスタ → 集積回路(IC)
- ・ メモリ素子(主記憶)
 - 水銀遅延線/放電真空管 → 磁気コア → トランジスタ → ICメモリ
- ・ 2次記憶装置
 - パンチカード → 磁気テープ・磁気ドラム → 磁気ディスク(FDD/HDD)・光学ディスク(CD/DVD/BD) → ICメモリ(SSD)
- ・ 入出力装置
 - パンチカード → タイプライタ → CRT/キーボード/マウス → 液晶モニタ・タッチパネル

「規模・速度・耐久性・価格・使い勝手」の面で急速な進歩
→ 計算機アーキテクチャの革新の原動力に

初期のメインフレーム

- IBM701(1952年):
 - 世界初のノイマン型の商用電子計算機
 - 「商用」とは、大学などの国家研究プロジェクトではなく、型番を指定してお金を出せば買える計算機
 - ただし、大企業や役所などでないと買えないくらいの価格
- 技術の進歩と市場競争で、次々に新機種が登場
 - ほとんどの場合、大企業や役所が基幹業務を行うために、計算機メーカーに発注して作らせて購入する形 → 競争入札
 - 同じ価格でより高性能に、同じ性能でより低価格に
 - 「メインフレーム」と呼ばれる計算機の系譜

休憩

- ここで、少し休憩しましょう。
- 深呼吸したり、肩の力を抜いてから、次のビデオに進んでください。

バッチ処理と初期のOS

コンピュータは高価で、
台数も限られている状況

- ・ ノイマン型計算機はプログラムを差し替えれば
様々な計算ができる → 多数のユーザが共用する
 - ユーザAのプログラムの計算が終わったら、ユーザBのプログラムの
計算を行う → 多数のユーザ(大学教授など)が順番待ちになる
 - 計算機センタに受付窓口を作って、オペレータの人がプログラム
(パンチカードなど)を預かって、計算結果が出たら呼び出して渡す
(→ 深夜に計算が終わる場合、オペレータがいないことも)

バッチ処理と初期のOS

コンピュータは高価で、
台数も限られている状況

- ・ ノイマン型計算機はプログラムを差し替えれば
様々な計算ができる → 多数のユーザが共用する
 - ユーザAのプログラムの計算が終わったら、ユーザBのプログラムの
計算を行う → 多数のユーザ(大学教授など)が順番待ちになる
 - 計算機センタに受付窓口を作って、オペレータの人がプログラム
(パンチカードなど)を預かって、計算結果が出たら呼び出して渡す
(→ 深夜に計算が終わる場合、オペレータがいないことも)
- ・ **バッチ (batch) 処理:**
 - 複数のプログラムを一括して磁気テープに格納しておき、自動的に
順次実行し、計算結果を別の磁気テープに自動的に書き込む方式
(→ オペレータは24時間待機しなくてよくなる)
 - オペレータの仕事を代行するプログラム(バッチモニタ)が常駐
→ オペレーティングシステム(OS)の原形
 - 主記憶容量が大きくなって初めて可能になった

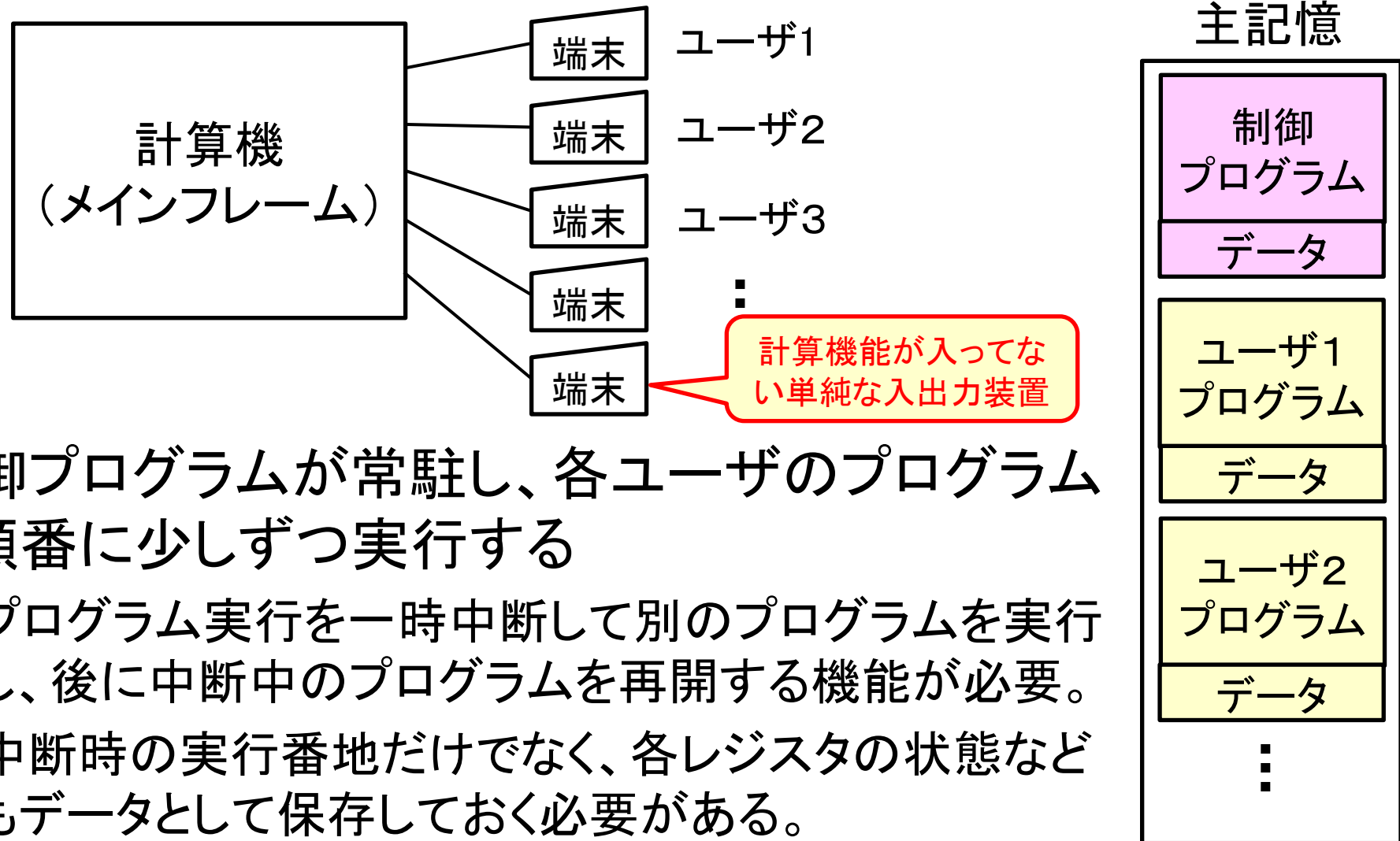
タイムシェアリングシステム

- ・ バッチ処理は、結果が出るまで時間がかかる
 - プログラムを窓口を持って行って、翌日に結果を受け取りに行ったら、「文法エラーで異常終了」で、書き直しになり、最初の1日が無駄になった、という話がよくある
 - 計算機を対話的に使えたと、早く仕事が終わるはずだが、当時の計算機は貴重で、1人で占有するわけには行かない
 - 実際、人間が考えたりキー入力する速度は、計算機の演算に比べてはるかに遅いので、計算機を占有するのはもったいない

タイムシェアリングシステム

- ・ バッチ処理は、結果が出るまで時間がかかる
 - プログラムを窓口を持って行って、翌日に結果を受け取りに行ったら、「文法エラーで異常終了」で、書き直しになり、最初の1日が無駄になった、という話がよくある
 - 計算機を対話的に使えと、早く仕事が終わるはずだが、当時の計算機は貴重で、1人で占有するわけには行かない
 - 実際、人間が考えたりキー入力する速度は、計算機の演算に比べてはるかに遅いので、計算機を占有するのはもったいない
- ・ **タイムシェアリング (time-sharing) システム:**
 - n 人のユーザに対して、(例えば)100分の1秒ずつCPU処理時間を割り当てて、順番に切り替えて実行すると、個々のユーザから見ると、あたかも計算機を1人で占有しているように見える。
 - 実際は n 本のプログラムを同時に主記憶に並べて、少しずつ切り替えながら、見かけ上は同時に実行している

タイムシェアリングとマルチプログラミング



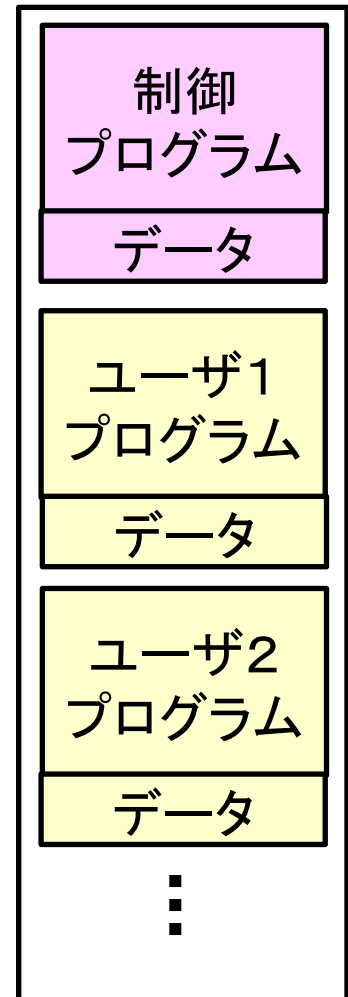
- ・ 制御プログラムが常駐し、各ユーザのプログラムを順番に少しずつ実行する
 - プログラム実行を一時中断して別のプログラムを実行し、後に中断中のプログラムを再開する機能が必要。
 - 中断時の実行番地だけでなく、各レジスタの状態などもデータとして保存しておく必要がある。
 - 主記憶容量と計算速度が大幅向上したため可能に。

マルチプログラミング向けの機械語命令

- ・ ユーザプログラムが置かれる場所(開始番地)が実行時によって変わる。
 - 間接アドレスやインデクスアドレス方式が有効
 - 相対ジャンプ命令
(現在番地よりn番地だけ前方/後方へ飛ぶ)
- ・ 各レジスタの状態を主記憶に退避する命令
- ・ タイマーによる割り込み

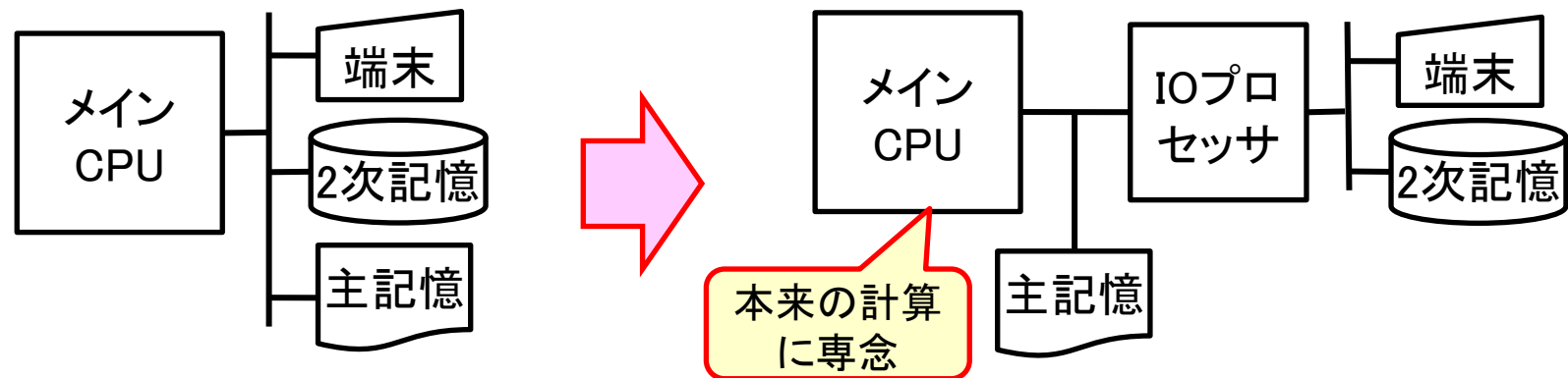
高機能な機械語命令があればプログラミングは容易になるが、ハードウェアは複雑化する。

主記憶



IOプロセッサとバス構成

- 初期の計算機は入出力処理も同じCPUで実行していた。
 - キーボード・プリンタ・磁気テープなどの入出力速度は、CPU本体の計算よりも何桁も遅い。→ CPUが待たされてしまう。
- 入出力プロセッサ (IOプロセッサ):
 - 貴重なCPU時間を有効活用するため、入出力専用のプロセッサを別に用意し、データをまとめて転送する方式が開発された。
 - データは主記憶の特定の領域 (バッファ) を通して受け渡される
 - バス構成も改良された。



上位互換性とファミリー思想

- ・ 技術の進歩により、数年ごとに新機種が登場し、計算機センターの設備を更新するたびに高性能機にグレードアップ
 - － 旧機種用に作成したプログラムが作り直しになると、とても困る！
 - － 競争入札の条件に入るようになった

上位互換性とファミリー思想

- ・ 技術の進歩により、数年ごとに新機種が登場し、計算機センターの設備を更新するたびに高性能機にグレードアップ
 - 旧機種用に作成したプログラムが作り直しになると、とても困る！
 - 競争入札の条件に入るようになった
- ・ **ファミリー思想**（例：IBM 360シリーズ 1960年代中頃～）
 - 上位互換性：グレードが上位の機種は、下位の機種用に作られた機械語をそのまま実行できる。（逆は必ずしも成り立たない）
 - 機械語の命令セットが包含関係にある
 - IOプロセッサや端末部品などもファミリー同士だと再利用しやすい
 - 設計が容易に
 - コネクタやバスの規格化・標準化が進む



黎明期～初期のアーキテクチャ技術の主目的

- ・ 計算機が使えることが貴重だった
 - 国家に1台とか州に1台とか。日本では例えば旧帝大に1台ずつ。
 - 使いたい人がたくさんいて、貴重なCPU時間を取り合っていた。
(例えば、CPU時間使用料1秒10万円で、1週間待ち)
 - 貴重なCPU時間を少しでも有効活用したいという要求
 - ユーザ(超エリート研究者)の時間も貴重なので有効活用したい

黎明期～初期のアーキテクチャ技術の主目的

- ・ 計算機が使えることが貴重だった
 - 国家に1台とか州に1台とか。日本では例えば旧帝大に1台ずつ。
 - 使いたい人がたくさんいて、貴重なCPU時間を取り合っていた。
(例えば、CPU時間使用料1秒10万円で、1週間待ち)
 - 貴重なCPU時間を少しでも有効活用したいという要求
 - ユーザ(超エリート研究者)の時間も貴重なので有効活用したい
- ・ 当初は、複雑な回路や大規模な主記憶が作れなかった
 - 不安定な真空管・トランジスタから、安定して量産可能な集積回路へ
 - 単純な命令セットから複雑な命令セットへ
 - IOプロセッサなどアーキテクチャの様々な工夫が可能に
 - 主記憶メモリは常に高価だが、使える容量は徐々に増大
 - バッチモニタやタイムシェアリングなど、ユーザを待たせないでプログラミングに集中させるための工夫が可能に
(主記憶容量が足りなければ無理だった)

今回のまとめ

機械語命令と内部動作(2)

- 主記憶アドレス参照方式

アーキテクチャの基本知識(1) (分類と概観・初期のメインフレーム)

- 価格・時代・用途による汎用計算機の分類:
 - メインフレーム、ミニコン・ワークステーション、パソコン
- 黎明期～初期のメインフレーム技術
 - 機械式から電子式へ
 - ハードワイアドとノイマン型計算機
 - バッチ処理とタイムシェアリング
 - IOプロセッサとバス構成
 - ファミリー思想