1　2012/05/19 qpstudy

1.1　だれでもわかるハードウェア：Akihiro Kuwano

大雑把な話をやります

1.1.1　自己紹介

- @kuwa_tw

1.1.2　コンピュータの５大要素

- 入力
- 演算
- 制御
- 記憶
- 出力

1.1.3　ノイマン型コンピュータ

- 今のコンピュータの基礎

1.1.4　コンピュータの基本構造

- 5 大要素が基本!!

■入力：**キーボード**

- ユーザからのデータ扱う

■記憶：HDD or SSD, **メモリ**

- 入力したデータを CPU に渡すためなど、置いておく。
- 補助記憶・主記憶

■演算：CPU

- 演算をする

■制御：CPU

- 各ハードウェアの動作を制御
- CPU と一緒になっている

■出力：ディスプレイ

- ユーザへの出力

1.1.5 業務へ

- ボトルネック調査
  - CPU
  - Memory
  - Disk IO
  - IO stat やらを見ます
- 効率の良いプログラム
- 効率の良いアーキテクチャ
  - 効率が良いとは... コンピュータにとって, アーキテクチャにとって良いということ

1.1.6 質疑応答

- ENIAC：最初のコンピュータっぽいやつ
  - 物理的な回路を組み替えまくってプログラミングする：ハードウェアコーディング
- トランジスタ：切り替えスイッチだよー
  - 分岐の処理とか
  - 昔は真空管だったよ
- ボトルネック調査の具体例とかは?
  - 入出力はコンピュータの性能とあまり関係がない
  - 単体を調べればいい場合：CPU, Memory, Disk
  - アーキテクチャを考える場合：CPU と主記憶をつなぐ部分, ディスクをつなぐバス部分 ...

1.2 CPU の話：sho kisaragi

CPU の動作を理解することで問題解析に役立てる

1.2.1 自己紹介

- @sho7650
- カレー・パヒューム・深田恭子
- CE -> IT スペシャリスト -> IT アーキテクト

1.2.2 目的

- CPU の振るまいと原理
- 問題の解析

1.2.3 コンピュータ

- 5 大要素
- ノイマン型アーキテクチャ：プログラム内蔵型 (?)
- Processor
- Programs
- Data

1.2.4 CPU

- CPU からの IO
  - I/O
  - Memory

1.2.5 CPU の動作：4 つの原則

- Fetch
  - 主記憶 (Memory) から Data をとってくる
- Decode
  - 主記憶から取ってきた命令を回路で実行できる命令に変換する
- Execute
  - 変換した命令を実行する
- Write back
  - (ものによって違うどこに返すかが違うが) 出力を返す

1.2.6 CPU Clock

- 水晶 (クオーツ) に電圧をかける
- 1 Clock ごとに動作が行われる
- 世界最速は 5.2 GHz：IBM

■Pipeline

- 1 Clock ごとに、それぞれ 1 つ (Fetch〜Write back) を行う
- Super Pipeline：fetch を 1 Clock

■Super Scalar

- 例えば、計算が全部足し算だったら一気にやればいい。

■Out-of-Order

- 処理に時間かかるものと、すぐ終わるものの順番を整理して、トータルで早くする。

例えば、5 個のタスクがあって、2 番目だけがメモリ読み出しだったとする。
メモリの内容が、５番目まで使われないなら、並列に 2 と 3,4 番目のタスクをやれば良い。

1.2.7 CISC vs RISC

- CISC : 回路を大きくする。命令豊富で何でもできるけど、1 Clock でできることが減る。
- RISC : 回路を小さくする。回路は単純で、命令を組み合わせて、トータルで 1 Clock になるくらい。
- 中は RISC で動いてるんだけど、外からの命令は CISC のように豊富にしてる?

1.2.8 Peak CPU Clocks

- 計算速度をあげる
  - 光の速さ
  - 高電圧をかける -> 回路中で干渉して、計算するのが大変!!

1.2.9 CPU **がどんな風に動くか**

- Z80 : 8 bit CPU で学んでみる

■What is bit width?

- あんまり定義がない
- 1 回の命令が 8bit とか。
- 命令セット 16 bit とかはありえる。
- bit 数が増える -> 1 回の命令で伝える内容が増える-> 早くなるハズ

■Machine Lang. やっぱりマシン語を学ぼう
Assembler Lang.

■Register vs Memory Register をどう使うか!? -> Assembler をやればわかる

- PC(Program Counter) が実行してる箇所

■Endian

- Big Endian : 順 (“ABCD” を ABCD でメモリに格納)
- Little Endian : 普通の順 (“ABCD” を CDAB でメモリに格納)
- とにかく、CPU によって格納のされ方がちがう

■A **レジスタ** (Accumulator)

- 必ずここを経由して計算を行う #### Status (flug) Register
- 計算の結果を反映する
- 計算結果で繰り上がりがあったら、それを格納したり、

- 0 になったら、0 ということを格納する。

1.2.10 Functions

- Data Transfer
  - Data Transfer(LD, PUSH, POP)
  - Exchange
  - Block Transfer
- Data Processing
  - Arithmetric Operation(ADD,SUB,INC,DEC)
  - Logical(AND,XOR,OR,CP)
  - Skew(RL,RR,SLA,SRA)
- Test and Jump
  - Jump(JP,JR,DZNJ,CALL,RET)
  - TEST の結果は flag に入ってその結果を見て、Jump する。
- Input/Output
  - Input
  - Output
- Control
  - NOP,HALT

1.2.11 Mnemonics and Operands

ex. Hello World

| Mnemonic | Operand | LD | DE,0C000H | LD | HL,#MSG LOOP: | LD | A,(HL) | OR | A | RET | Z | LDI | | JR | LOOP ...

1.2.12 Summary

- Assembler
- English
- いくつか違ってるものがあるかも。また、解釈の差があるかもしれません。
- 今日出たキーワードにつてい 自分で調べてください。

1.3 コンピュータアーキテクチャ I/O 入門 : hasegawa さん

1.3.1 自己紹介

- @hasegaw
- Xen/KVM, FreeBSD virtio など
- 著書: いっぱい

1.3.2 目標

- コンピュータにおける I/O
- IA-32 での I/O の種類
- 入出力デバイスの種類など

1.3.3 入出力とは何か

- I/O はコンピュータ (計算をするため) にとっては必須ではない

1.3.4 入出力の基本

- I/O ポート
- 割り込み
- Direct Memory Access
- メモリマップド I/O

1.3.5 I/O ポート

- コンピュータ出利用される古典的な入出力 例) ポート 60 番にデータを送ると、その先のデバイスにデータが届くポート 60 番からインプットとかもできる。
- プロセッサから外部に接続されるためのデジタルインターフェース
- 1 アドレス 8 ビット x64K, ON(1) or OFF(o) をつう有心
- 基本的に 8bit 単位なので低速
- 古いキーボード制御など

1.3.6 割り込み

- デバイスから CPU へイベントを通知するための信号
  - IRQ 割り込み入力用の 16 本の信号線
- 割り込みハンドラ
- 例えばキーボードの状態が変わると、イベントが起きて、割り込みが発生するなど。

1.3.7 16 ページ (資料公開待ち)

- int09_handler : 割り込みハンドラキーが押されてないかのチェックをしたりする。 _int09_function とかで読んでる?

1.3.8 入出力装置をどのように使うのか

- 高校の頃?のライト点灯システムの実装例

1.3.9 メモリマップド I/O(MMIO)

- プロセッサの物理メモリ空間に、デバイス上のメモリ空間をマッピングする。
- ソフトウェアからデバイス (普通のメモリに限らない) メモリにアクセスできる

1.3.10 ダイレクトメモリアクセス (DMA)

- プロセッサの指示にしたがって DMA コントローラがデータ転送する
- データ転送の開始時/終了時のみプロセッサが干渉するため、プロセッサが時間を節約できる。(他の仕事もできる)

1.3.11 IRQ と MSI

- IRQ の数には制限がある
  - インテルの現在のものだと、たかだか 15 本しかない
  - 他の目的にも使われる部分があるため、複数デバイスで共有するしかない。
  - 共有してるとどのデバイスが割り込みを受けたかわからないかもしれない-> デバイスがフラグを持っていて、それを使って区別している。
- キーボード、マウス、タイマ、....

1.3.12 IRQ の共有

- 割り込みが発生したデバイスのフラグが 1 になる。
- 割り込みが発生したら各ドライバのステータスをチェックし、必要なハンドラのみが実行される。

1.3.13 バスとはなにか

- 内部バス：CPU 内部
- 周辺回路：
- 拡張バス：外部デバイスとの接続

1.3.14 PCI バス

- コンピュータを各種ハードウェアデバイスに接続するための標準仕様
  - 32bit パラレル通信, 33MHz, 帯域幅 133MB/s
- プラグアンドプレイ
  - 挿したら勝手に設定してくれる、とかいう方の意味

1.3.15 PCI-X バス

- PCI バスの上位互換
- 周波数とか、帯域がグレードアップ

1.3.16 PCI Express

- 複数レーンを束ねて使う
- シリアル転送になった!!!! && スピードアップ!!!!
- 1 レーンで 250MB/s
- 4 レーンで 1 GB/s, 16 レーンで 4GB/s
- 活線挿抜に対応
- ソフトウェアレベルでの PCI 上位互換

* * 25GHz の周波数でデータをシリアル伝送 (1Clock 当たり 2 ビット伝送)

1.3.17 PCI Express 2.0

- クロック 2 倍!
- 16 レーンで 8GB/s の帯域をサポート
- IRQ の代わりに... Message Signal Interrupt のサポートが必須に！

■skew とは

- まず PCI Express は複数レーンを使っている。
- 各レーンごとに速度差があるかもしれないので、適当に同期を取る & データ順を考慮して復元しないといけない

1.3.18 Message Signal Interrupt(MSI) <- **この変更は仮想化環境に大きな影響があるに違いない!!パフォーマンスが上がる。**

- メモリ書き込みにより、割り込みを通知する
- PCI デバイス利用する
- IRQ を使わない
- コントローラが仮想的に普通と違うメモリ書き込みをしたら、CPU がフラグをチェックせずとも、どのデバイスに割り込みが発生したかわかる！
- 32 メッセージまで共存可能
- Enhanced MSI(MSI-X)
  - PCI Express バス向けの MSI
  - PCI Express 2.0 では実装必須
  - 2048 メッセージまで共存可能

1.3.19 CPU **間のインターコネクト**

- マルチコア
- マルチプロセッサ
  - UMA：対称型マルチプロセッサ
    - どの CPU からでも同じ時間でアクセスできるが、全 CPU でバスを共有する

  - NUMA：非対称メモリアクセス
    * 近いデバイスには早くアクセスできるが、遠くのデバイスにはアクセスに時間がかかる
    * バスの共有しない (?) か共有が少ないのか
    * メモリとか増設するときにも、マルチプロセッサ CPU だとどれかの CPU に近いやつとかあって気をつけよう。
    * 場合によっては遅くなる。チューニングを検討しよう
- QPI (AMD)
- HyperTransport (Intel)

1.3.20 まとめ

- I/O とは
  - コンピュータのに対するデータ入出力の機能
- 代表的な I/O 方法
  - I/O ポート, 割り込み
  - メモリマップド I/O.DMA
- バス
  - 拡張バス
    * PCI, PCI-X, PCI Express ...

1.3.21 I/O を勉強するには

- 超初心者
  - Arduino
  - 本もいっぱい, キットも買える
- 初心者
  - PIC
  - USB マイコンボード とかも売ってる
- もっとガチな人
  - BeagleBoard

1.4 インターフェース入門 shinonome

1.4.1 自己紹介

- @H_Shinonome

1.4.2 資料が 100 ページあるらしいので最初から挫折しておいた。

1.4.3 RS-232C

- 25 pin
- 9 pin のやつは標準規格に準拠してない。同期できないし。

- 同時送信数：1

1.4.4 DB-60：Cisco とかでよく見る

- 60 pin
- 115200 bps
- 同時送信数：1

1.4.5 どっちも同時送信数 1

- 端子数多いのに...
  - 直接データ送受信しない端子
  - 同時に同じデータを複数端子で送る
  - そもそも使ってない端子もある
- 信頼性の向上のため

1.4.6 動作周波数

- 33MHz => 1 cycle：30 nano sec
- SDR：Single Data Rate
- DDR：Double Data Rate

1.4.7 通信速度は頭打ち

- 費用対効果
  - 半導体性能

1.4.8 IEEE 1284

- 25 pin
- プリンタポートとして出たものを拡張した規格
- 同時送受信 8 bit

1.4.9 SCSI

- 50 pin
- 同時送受信 8 bit

1.4.10 ATA

- 40 pin
- 同時送受信 16 bit

↑ パラレルが抜いた...

1.4.11 **なぜシリアルに戻るのか**

- 半導体性能アップ
- スキュー

1.4.12 USB (Universal Sirial Bus)

1.4.13 FireWire (IEEE 1394)

1.4.14 S-ATA

- 端子数: 7
- 通信速度: 1.5 Gbps - 6 Gbps

1.4.15 SAS

- 端子数: 7
- 通信速度: 1.5 Gbps - 6 Gbps

1.4.16 Thunderbolt

- 端子数: 20
- 通信速度: 10Gbps * 2
- 非同期パラレル通信

1.4.17 InfiniBand

- 端子数: 4-48
- 通信速度: 2Gbps - 163.64 Gbps
- シリアル (1X), 非同期パラレル (2X - 12X)

1.4.18 **最近の流行**：**非同期パラレル**

1.4.19 **最後**

左: SAS 右: SATA

- 違いを考えよう!!

1.5 **ニフティの宣伝**

1.5.1 Nifty Cloud C4SA

- 5 月末クローズドβ
- APaaS + ブラウザ上の IDE + ソーシャル開発 的な感じ