Rechnerarchitektur

Zusammengetragen vom Marc Landolt

Die verschiedenen Betrachtungsebenen

- Rechnerebene
- Hauptblockebene
- Registertransferebene
- Schaltwerkebene
- Ebene elektrischer Schaltungen
- Physikalische Ebene

1 Blockschaltbild eines Rechners

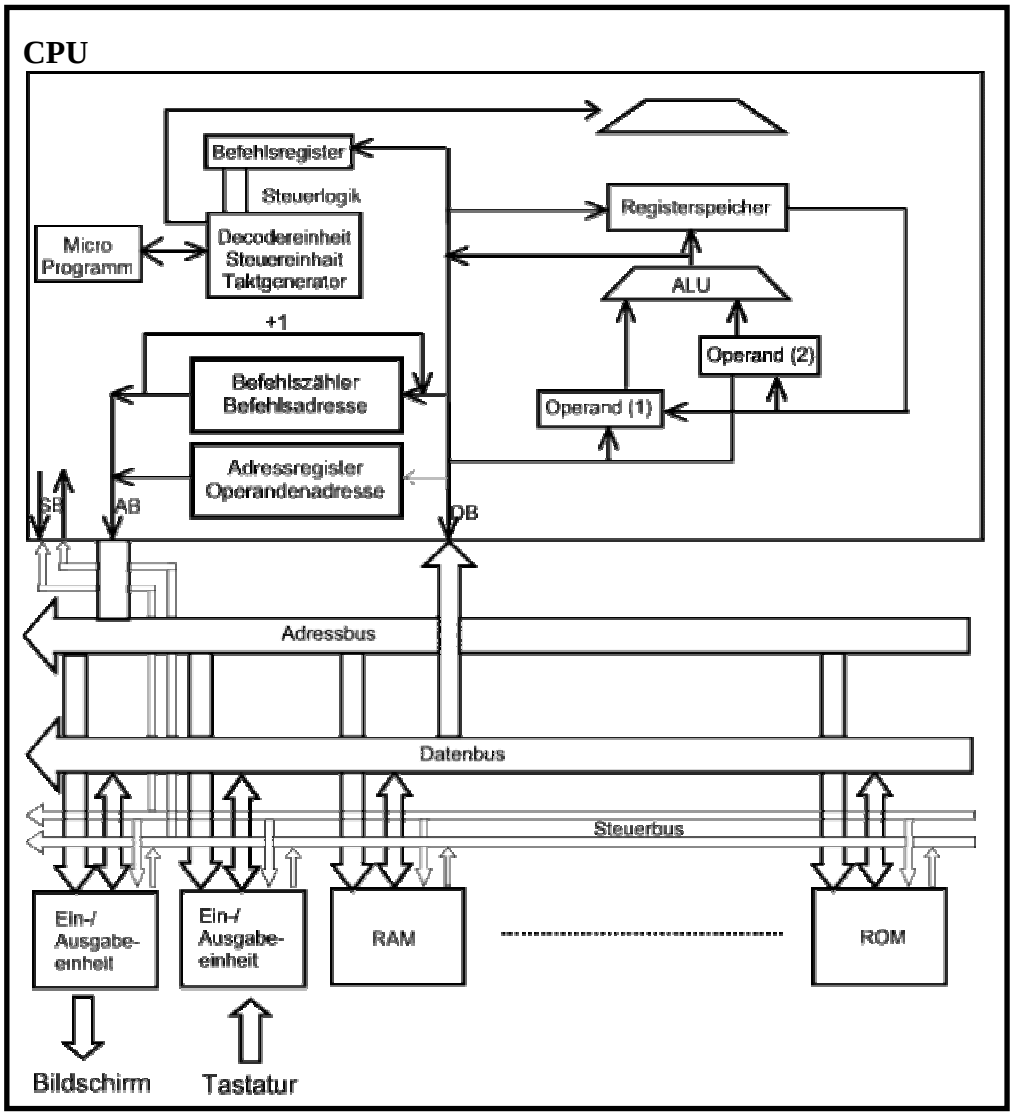


Abbildung 1[1]

[1] http://www.bergers.co.at/sites/hardware/pic/cpu_big.gif

2 CPU

Die Zentrale Recheneinheit (**C**entral **P**rocessing **U**nit) ist der zentrale Ort des geschehens. Sie besteht aus:

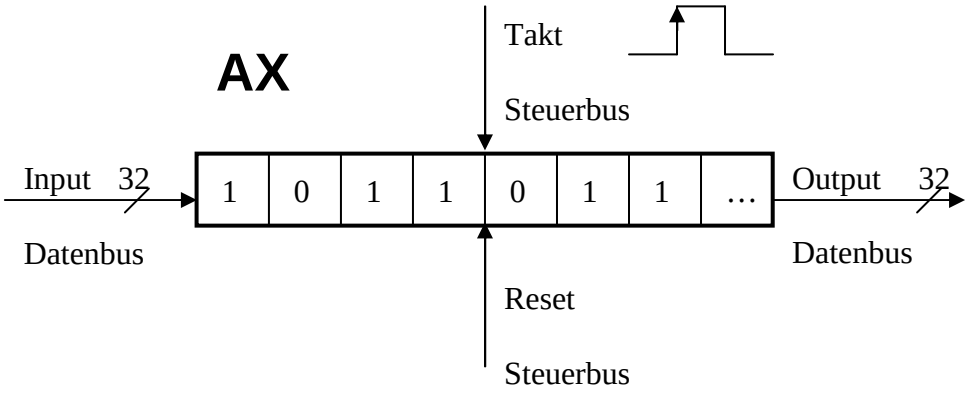
- ALU (Arithmetic Logic Unit = Arithmetische Logische Einheit)
- Befehlsregister
- Statusregister
- Register
  - Datenregister
  - Adressregister

  } General Purpose Register (GPU) {AX, BX, CX, DX}
  - Status Register (Flag register) {F}
    - Z (Zero)
    - C (Carry)
    - N (Negative)
    - V (Overflow)
    - I (Interrupt)
    - P (Parity)
    - …
  - Floating point Registers
  - Contitional Registers (True / False)
  - Constant Registers (Zero, One, Pi…)
  - Vector Registers
  - Special purpose Register (Befehlspointer = Instruction Pointer {IP})
  - Machine specific regsiters
- Cache

DETAILS COMING SOON

2.1 Das Register

Das Register besteht aus Vorderflanken gesteuerten D-Flip-Flops. Es können darin Binärworte gespeichert werden. Im Allgemeinen sind sie so breit wie die Architektur der CPU. Entsprechend der Busbreite der CPU wird die Busbreite des Systems gewählt. Ausnahme war 386 AX, dort wurde, da die Industrie noch auf 16 Bit eingerichtet war z.B. das CL Register auf den Bus gegeben, das heisst intern wurde 32 Bit gerechnet, jedoch musste, sobald der Bus des Systems angesprochen werden sollte auf 16 Bit „umrechnet“ werden. Die Vermutung liegt nahe, dass diese suboptimale Lösung nicht die eines Technikers sondern eines „Marketing-Fritzen“ war.



Es können Werte vom Datenbus übernommen werden, bzw. auf den Datenbus gelegt werden. Dies funktioniert einfach gesagt so:
Eingabe: Im Grunde stehen die Werte des Datenbusses immer am Register an, da dies die Werte aber nur dann übernimmt, wenn der Takt gerade von Null auf Eins springt, wird nur dem Register den **Takt** geben, welches den Wert übernehmen soll.
Mit dem **Reset** wird das Register auf einen festen Wert zurückgesetzt. Was unter Anderem z.B. wichtig für das IP Regester während des Bootvorgangs.
Weiter ist zu sagen, dass aus Gründen des Timings meist nicht die Taktleitung verwendet wird, sie sollte mit möglichst wenig Logik ausgestattet sein. Stattdessen wird über eine zusätzliche **write enable** Leitung geregelt. Der Wert muss eine bestimmte Zeit anliegen, vor dem Lesen muss der wert die so genannte **setup time** anstehen, dann wenn das Register die Werte zu übernehmen beginnt muss das Signal für die **hold time** anstehen. Wonach der Wert nach einer typischen Zeit, dem **propagation delay**, am Ausgang ansteht.
Um nun, sagen wir mal 4, Register mit der Breite von 1 Bit so miteinander zu vernetzen, braucht es bereits n(n+1) Verbindungen, vor jedes der n Register wird ein Multiplexer so geschaltet, dass er alle Ausgänge der Anderen Register wählen kann, so entstehen dann bei 4 Register 20 Leitungen. Um das ganze mit weniger Leitungen machen zu können hängt man an jeden Registerausgang ein 3-state Gatter, welches folgende zustände annehmen kann:

EN	IN	OUT
0	0	Hochohmig
0	1	Hochohmig
1	0	0
1	1	1

Die Ausgänge der 3-state Schaltungen gehen dann auf eine Busleitung wo ein anderes Register dann den Wert wieder holen kann, dies geschieht über eine Write Enable (we)

Tri-State

Ein 3-State Schaltung ist im Grunde eine Gegentakt Schaltung, mit einer Logik. Sie hat 3 Zustände: Hochohmig, wenn das EN auf Null ist, sonst hat sie den Wert des Eingangs (IN). Dabei ist zu beachten, dass wenn aus Versehen zwei 3-State gleichzeitig auf den selben Bus einen unterschiedlichen Wert geben zerstört dies den oberen Transistors Q1 des 3-States mit dem Wert 1 und/oder Q2 des 3-States mit dem Wert Null.

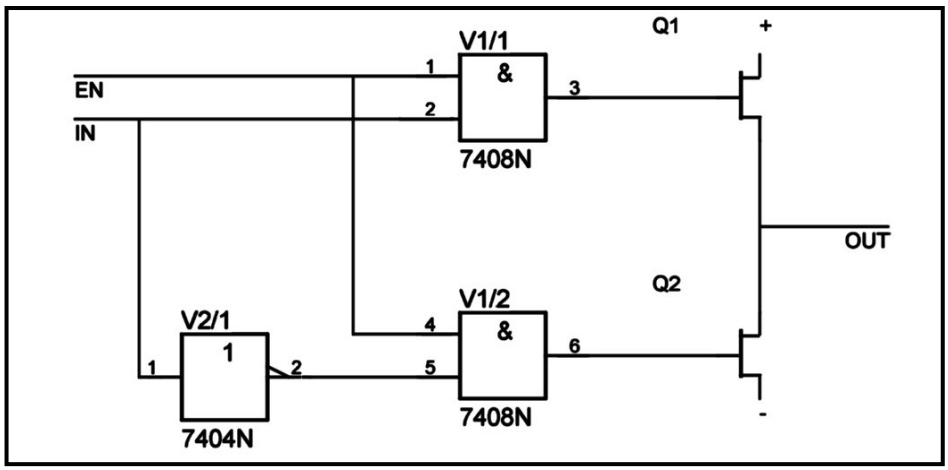


Abbildung 2[2]

Jedes Register hat einen eineindeutingen (Bijektiv) Namen und sind somit anders als das RAM direkt ansprechbar, z.B. der Befehl Mov AX, FFh füllt das AX mit Einsen. Die ALU (Arithmetische Logische Einheit) kann nur mit den Werten in den Registern rechnen. Z.B. ADD AX, BX addiert den Inhalt des BX-Registers zum AX Register hinzu. Somit stehen die Register im Brennpunkt des Geschehens.

2.2 Mnemonics und Opcode

Im Prinzip könnte man direkt so Programmieren:
10111011 11111111
dies würde jedoch wohl kein normaler Mensch mehr verstehen, deshalb hat man die so genannten Mnemonics erfunden. Mnemonics sind Abkürzungen, die einfach zu merken sind: **MOV AX, FFh** heisst zum Beispiel, lade das AX Register mit dem Wert FFh. Was nun der Assembler macht ist eigentlich nichts anderes als in einer Tabelle nachzuschauen was MOV AX für einen so genannten Opcode (OPeration CODE) hat. Also welches Binärwort der Prozessor benötigt um einen Wert ins AX Register zu laden. Der Prozessor nimmt nun dieses Binärwort und vergleicht es mit seinem Befehlsregister (Microcode) und sieht, dass er das AX Register laden muss.

Beispiel:
Das Befehlsregister (Instruction Pointer / IP) zeigt auf einen Speicherbereich im RAM. Der CPU holt sich den Wert aus dem Speicher, vergleicht ihn mit dem Befehlsregister, und sieht dort, dass es der Befehl für das Laden des AX Registers ist (MOV AX) , somit weiss die CPU auch, dass sie noch einen weiteren Wert benötigt, um den Befehl auszuführen und holt sich

[2] genaueres unter: http://www.princeton.edu/~wolf/modern-vlsi/Overheads/CHAP7-2/sld032.htm (11.2006)

diesen Wert, in unserem Fall ist das z.B. FFh. Nun hat sie die nötigen Informationen um den Befehl ausführen zu können und lädt FFh ins AX Register. Danach wird IP so erhöht, dass er auf den nächsten Opcode im Speicher zeigt und nicht etwa auf einen Datenwert, denn dieser würde vermutlich zu einem Absturz führen. In unserem Fall wird also das IP um 2 erhöht, wäre der Befehl NOP also keine Operation würde auch kein zweiter Wert benötigt, dies hiesse, dass der nächste Befehl gerade in der nächsten Speicherstelle liegt, womit der IP nur um 1 erhöht werden müsste.

Opcodes können ein 2 Verschiedene Arten unterteilt werden

- Steuernachricht
- Operatoren Nachricht

3 Das RAM (*R*andom *A*ccess *M*emory)

Wobei zu sagen ist, dass RAM nicht für Zufallsspeicher steht, denn im Computer sollte nichts dem Zufall überlassen werden. RAM bedeutet beliebig zugreifbaren Speicher.
Im Prinzip besteht RAM auch aus Flip-Flops, mit dem Unterschied, dass die für alle Speicherstellen im RAM nur ein Anschluss vorhanden ist, somit muss es einen Weiteren Anschluss geben um die Speicherstelle im RAM auszuwählen, auch muss unterschieden werden, ob aus dem RAM gelesen oder ins RAM geschrieben wird.
Somit hat das RAM folgende Anschlüsse:

- Daten z.B. 16 Bit
- Adresse z.B. 16 Bit
- Takt (beim Synchronen) ???
- Reset (Für alle Speicherstellen) ???
- Read/not write bzw. Write Enable (WE)
- Output enable (OE)
- Power
- Ground

TODO:
Boolsche logik
CLK
Carry
Cisc
Risc
Butterfly check A butterfly test is basically a random seek test.
Asynchron / Synchron (mit takt des systems)
SISD
Memory management unit
Statisches ram
Dynamisches Ram (refresh)