Computermemory meets (beats) Intelligence

Overview of Memory development in computers and influence to APL programming

Gerald Dittrich

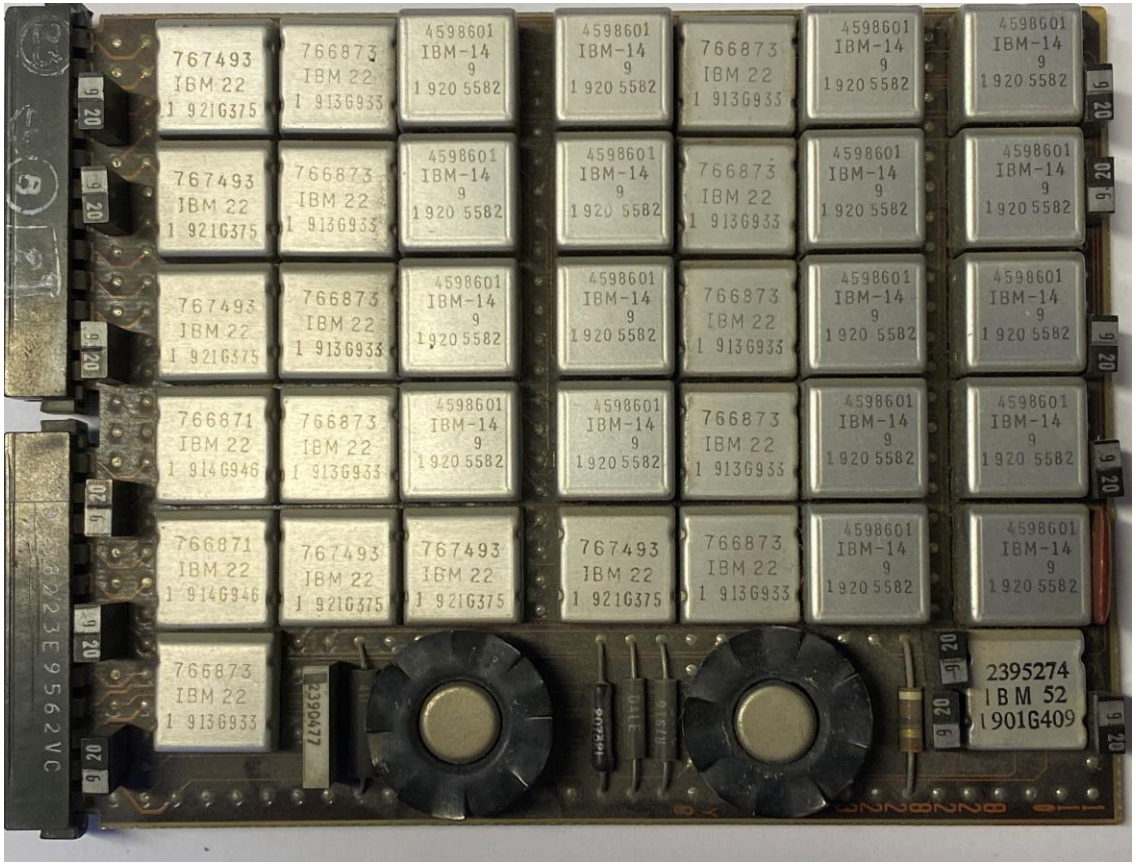
APL-Germany meeting

Berlin November 22-23, 2021

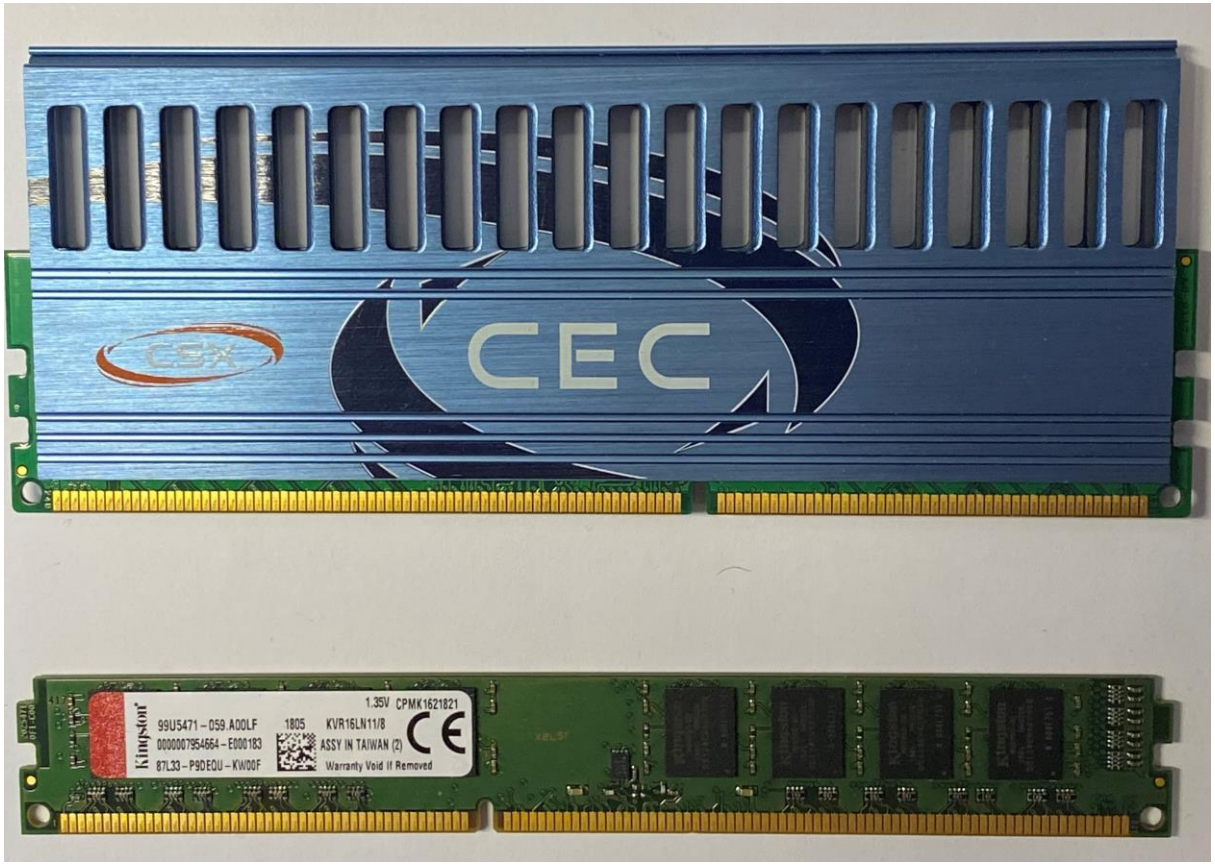
Memory in computers increased rapidly ??

- Examples (incomplete):
- 1975: IBM 5100 up to 64K memory
- 1976 : IBM APL-SV system Hamburg up to 200KB
- 1976 : IP-Sharp up to 512K
- 1977 : IBM CMS System Düsseldorf 3MB real memory up to 16MB virtual
- 1981 : IBM-PC 64K up to 640K with later IBM-AT
- 2010 : many PC‘s with motherboards up to 8GB
- 2016: Apple Imac with 32GB on board
- 2021: Dell Gaming Alienware PC with up to 128GB and 24GB GPU

Picture of IBM5100 16KB memory



Memory with 2GB, memory with 8GB



Usage of Core memory in some APL-Interpreters

- Some examples (from my “core memory”, data not proved , not complete)
- 1975 : IBM 5100: 58K (64K minus 6K for APL-emulator)
- 1976 : IBM CMS (3MB real core, 16MB virtual), IBM MVS and IBM VSPC – Systems
- 1982 :Example: IBM VSPC System Allianz Stuttgart (100KB for normal user, 200 KB with special allowance from the board of management, guys in my workshop for professionals got tears in the eyes when I told them that I have been working with up to 16MB since 1976),
- 1983 : IPM-PC up to 600K (max. 640K minus 40K APL-interpreter and aux-processors)
- 1988: IBM-APL for OS2 (up to ??)
- IBM APL for Windows (32-bit architecture) up to 1.5GB
- Windows APL-Plus (32-bit architecture) up to 1.2GB
- Dyalog APL (32-bit architecture) up to 1.5GB
- Dyalog APL (64-bit architecture) much more greater than 128GB (should be some 4.6E18, can’t be proved)

Influence of core memory to APL-programming

- Worksspace full (sword of Damocles):

- In the very beginning, lots of (unnecessary) loops hindered usage of APL:
- Problemsolving influenced by insufficient memory
- Blocklooping due to large database on disk storage
- Cutting matrices into small slices to avoid WS FULL
- Unable to use efficient APL-Functions (i.g. inner and outer product, indexing, encode and decode)
- Copying functions and variables from other workspaces
- Simulations influenced by small workspace (swapping needed)
- Growing database suddenly leads to WS FULL (reserved space might help to estimate how near WS FULL might be)

Use of encode (my favorite function!) is very efficient, but may lead to excessive need of workspace

- Idea behind encode (my favorite function):
- Produce all possible combinations to solve a problem
- Get rid off all combinations that obviously do not meet restrictions
- Prove remaining data if there are solutions to problem
- Example: game dice with 3 cubes (statistics behind probability to get 10)
- Examples: Aspirin problem (looking for all words containing the same characters as in “ASPIRIN“)
- Combination problem („LYNDON“ x „B“ = „JOHNSON“)
- Hunting crazy idioms (generating lots of mostly senseless combinations, proving if executions works and if execution is possible if result is correct)
  Example: 3 character idiom to answer the question if the first number in a numeric vector is the largest and only the largest

Example of ASP program

```
ERG←ASP STR;A;I;M;MA;R;R1;RED;STR1;Z
⍝o DPC/GD,24.12.2017
⍝p GENERATE ALL COMBINATIONS OF A GIVEN STRING STR
⍝r A2 ASP A1
⍝d
⍝c
⍝t APL2 VERSION
⍝s ⎕REF
⍝e1 STR←'ASPIRIN'
R←⍴STR
R1←⍴STR1←STR[2 ⎕REF STR] ⍝ ALL DIFFERENT OCCURENCES OF CHARACTERS IN STRING
Z←0,¯1↓⍳R*R
M←1+(R⍴R)⊤Z
A←+⌿STR∘.=STR1 ⍝ NUMBER OF OCCURENCES OF A CHARACTER IN STRING
M←(~∨⌿M>⍴A)/M ⍝ REDUCE M TO MAX DIFFERENT CHARACTERS
MA←STR1[M]
I←0
LI:→LIE×⍳R1<I←I+1
RED←A[I]=+⌿MA=STR1[I] ⍝ REDUCE TO NUMBER OF OCCURENCES OF A CHARACTER IN STRING
MA←RED/MA
→LI
LIE:
MA←⍋MA
MA←MA[2 ⎕REF MA;] ⍝ SORT AND ELIMINATE DOUBLE OCCURENCES
ERG←MA
```

Use of ΔREF

```
x← 2 ΔREF'ASPIRIN'
x
1 4 7 3 5 2
'ASPIRIN'[x]
AINPRS
```

Example where growing database leads to WS FULL

Decision 1: Plug in more memory

Decision 2: Change your program and waist a lot of time

- The Problem : Lyndon x B = Johnson (Combination problem discussed in 1971)
- Between 1963-1969 Lyndon Baynes Johnson was President of the United States of America

- How to solve the equation LYNDON x B = JOHNSON ?
- Solution is time consuming (not really difficult but boring) with lots of if-thens decisions
- Solution found in the internet (discussed in my speech from 2018)

Problem: The LYNDON x B = JOHNSON

- All solutions of the LYNDON x B = JOHNSON problem (can be solved in some seconds on an Intel 9 based computer (4 seconds for one character) but there is only one solution
- LYNDON x B = JOHNSON
- Result: 570140 x 6 = 3420840
- Result equal for all characters not in LYNDON JOHNSON
- No solution possible if character B occurs in LYNDON JOHNSON

Another example:

- MORTEN x ? = KROMBERG
- Only one possible solution if ? = K must be 0 !
- Result: MORTEN x B = KROMBERG
- 693517 x 8 = 03948136

APL-Solution of this Problem

- LYNDON x B = JOHNSON (written first in IBM-APL)
- MORTEN x ? = KROMBERG (written first in IBM-APL)
- Memory was sufficient to solve Problem
- Getting more curious for this challenging problem ?!
- What about NAPOLEON x ? = BONAPARTE
- A solution in principle should be possible (only 8 of 10 characters in forename and family name, forename: 8 characters, family name: 9)
- But: getting WS FULL (shame on IBM-APL !)
- Not willing to change my program and waisting time
- Solution: Plugging in 24GB (only 3 frames of 8GB) and changing to Dyalog-64bit-APL
-

A mathematician also is happy to prove that there is no solution!

- The dyalog 64 bit APL-Version (really no afford to migrate from IBM-APL version) worked well without WS FULL in a workspace of nearly 24GB (I did not prove how many GB really were needed)
- I was not sad that my program told me that there is no possible solution
- Glad of having solved my problem without changing my basic solution

- By the way: a former IT-Manager (Dietmar Freigang) of Allianz Life insurance Stuttgart (perhaps by far the biggest user of APL in Europe) advised me when discussing about optimizing programs (runtime and memory): Solving problems with additional hardware (if possible and not too expensive) might be much more better(and even cheaper) than changing your original program! (Don‘t spill but block). This statement still is true!!

Aspects to future: Can APL‘ers be happy?

- Nearly unlimited workspace allows integrated programming
- Unlimited workspace does not veil our view to problem solving (working straight forward without thinking about restrictions)
- Tempting use of powerful functions and operators is making happy

- But!
- Some functions and operators lead to exponential or semi-exponential rise of computing time (outer product, indexing, sorting)
- Result: Database 100 times larger: Computing time might be up to 10000 times higher
- Need for extreme fast CPU’s, parallel computing, usage of GPU’s, (quantum computers ???)

Using multiprocessor capabilities and large memory

- Intel i5: up to 4 cpu‘s (2 virtual) (up to 4 GHz)
- Intel i7: up to 8 cpu‘s (4 virtual) (up to 4 GHz)
- Intel i9-10900KF: up to 10 cpu‘s (the same virtual) (up to 4 GHz)
- AMD RYZEN 9 5950X , 16-Core (the same virtual), 72MB Total Cache, Max Boost Clock of 4.9GHz
- My favorite: Dell gaming computer Alienware with AMD RYZEN 9 5950X, 128 GB core memory, GPU Nvidia Geforce RTX 3090 (24GB), m2 disc (2 Terabyte), extremely fast Data transfer (6 GB/s), Dell curved wide screen display (4K) ……..

- More possibilities ?

Use as much memory power that you can get (ideas , suggestions))
(

- Distribution possible if problem can be partitioned:
- Distribute workload to available cpu‘s (do not use virtual cpu‘s : that may slow down computer speed due to swapping and partage memory!)
- Distribute workload to available (internet) computers
- Distribute workload to a cascade of computers
- Use GPU‘s (graphic processing units)
- Use m2 discs (example samsung 2 terabyte) as virtual extension of core memory
- More possibilities ? (ideas from the audience!)

Switching from one APL-Version to another

- Relation Host-APL to PC-APL
- Reasons to switch: No future aspects (no real updates, only correction of errors)
- No reaction to user requirements (IBM, rejected or long range consideration, e.g. 64 bit APL for Windows, extension of symbol table on Host-APL)
- Licence policy of APL-suppliers (example APL-PLUS –insurance company – 2000 licencies for consulting software to sell insurance offers in a bank)
- Need to change due to errors (SYSTEM ERROR)
- Need to change due to workspace limitations (WORKSPACE FULL)
- Great difficulties to go back (need for cover functions, especially when using operators that do not exist in the other APL)

Ring Parabel from Nathan the Wise (Lessing)

- Comparison of 3 APL-Interpreters with the 3 rings in the Ring Parable
- APL-Conference 2010 Berlin (Germany)
- Chairmans introduction (Gerald Dittrich)
- The right ring perhaps got lost?

Ring Parable

- **I quote the following briefly from Wikipedia :**
- ***The centerpiece of the work is the Ring Parable (German: Ringparabel), narrated by Nathan when asked by Saladin which religion is true: An heirloom ring with the magical ability to render its owner pleasant in the eyes of God and mankind had been passed from father to the son he loved most. When it came to a father of three sons whom he loved equally, he promised it (in "pious weakness") to each of them. Looking for a way to keep his promise, he had two replicas made, which were indistinguishable from the original, and gave on his deathbed a ring to each of them.***

Ring Parable

- ***The brothers quarrelled over who owned the real ring. A wise judge admonished them that it was impossible to tell at that time – that it even could not be discounted that all three rings were replicas, the original one having been lost at some point in the past – and that, to find out whether one of them had the real ring, it was up to them to live in such a way that their ring's powers could prove true, to live a life that is pleasant in the eyes of God and mankind (rather than expecting the ring's miraculous powers to do so).***
- ***Nathan compares this to religion, saying that each of us lives by the religion we have learned from those we respect.***

Some beautiful pictures of the greek mythology

Paris (Mythologie) – Wikipedia
de.wikipedia.org

Helena von Troja von A...
alamy.de

Helena (Mythologie) – Wikipedia
de.wikipedia.org

My favorite APL?

- Greek mythology: War of Troy
  Paris had to judge which of three godesses (Athena, Artemis, Aphrodite) is the most beautiful one – but Paris of Troj had been corrupted by Aphrodite, who promised him the most beautiful woman of the world
- But Helena was married with Menelaos, King of Sparta in Greece
- Result: 10 years war between Greece and Troj – it was a brutal economic war! 10 years of war between the gods (those who helped the greek warriors and those who helped the trojan warriors)
- My favorite APL ? It depends on …..

APL Situation in the past, present and future

- The past: I have been talking about usage of memory working with APL in the last 40 years
- Past: Difficulties arousing from not enough memory
- The present: Memory has grown tremendously (from 54KB to 128 GB) until now
- The future: Morten Kromberg is going to tell you something about the future of APL with DYALOG
- My internetaddress : gerald.dittrich@dpc.de please send me a message if you are interested in this demoworkspace (it is a dyalog workspace – fast written and tested, but with comments and not really dirty code, free for personal use) – I will mail it to you!