36-bit computing at DEC

Angelo Papenhoff/aap (pdp-6.net)

2018-07-15

Who am I?

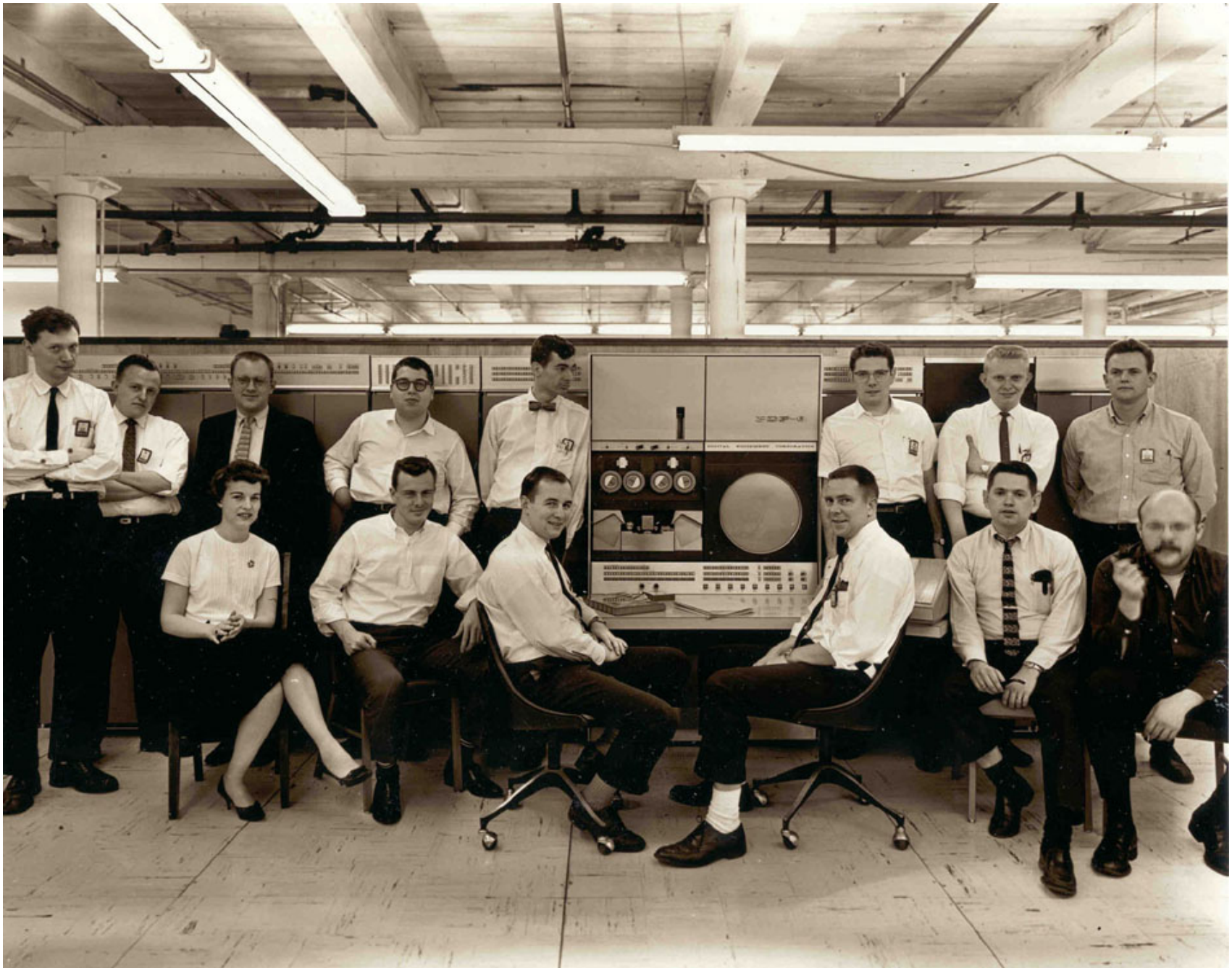
- got interested in the history of UNIX
- UNIX ran on a PDP-11 (a nice machine)
- the “11” implies there must more PDPs
- learned about the mythical PDP-10
- got to touch and play with a PDP-10 panel!
- Steven Levy’s book ”Hackers” did the rest

PDPs

- PDP-1: 18 bit, 4k, inspired by the earlier TX-0
- PDP-2: never built (24 bit PDP-1?)
- PDP-3: designed; built once by customer (36 bit PDP-1)
- PDP-4: 18 bit, 8k: new design, trade one opcode bit for one address bit
- PDP-5: 12 bit, 4k: condensed PDP-4
- PDP-6: 36 bit, 256k: different design, fast ACs, indexing
- PDP-7, -9, -15: PDP-4 architecture
- PDP-8,-12: PDP-5 architecture
- PDP-10: PDP-6 architecture
- PDP-11: 16 bit, 64k, byte addressed, 8 registers
- PDP-13: not designed
- PDP-14: not like the others

PDP-6: Development

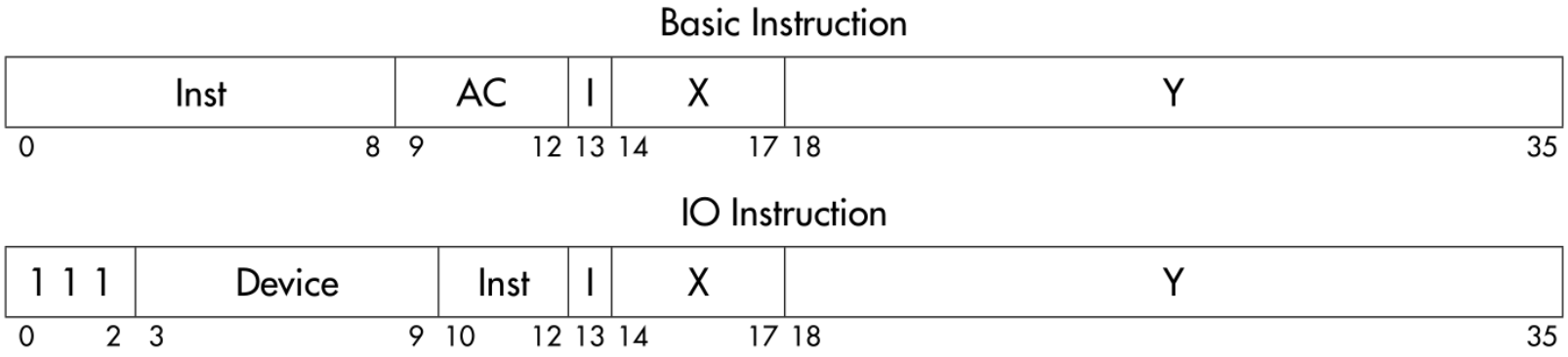
- DEC wanted to build a bigger machine
- Gordon Bell: architect
- Alan Kotok: assistant (later chief engineer of the PDP-10)
- many others…
- input from MIT hackers:
- a platform on which to implement LISP
  → needs space for two addresses per word
  → grows from 24 to 36 bits
- all previous PDPs accumulator machines:
  Peter Samson proposes accumulators/index registers
- brochure from 1963 describes older design
  PDP-6 is recognizable but different



PDP-6: Architecture

- 36 bit words
- 18 bit address space (256kw, “moby”)
  $\rightarrow$ two addresses per word (for LISP)
- low 16 memory locations are fast memory:
  accumulators, index registers, memory
- user and executive mode
- memory relocation and protection in user mode
- special IO instructions
- manipulate parts of words (bytes) with special byte pointers
- regular instruction set: lots of nonsensical combinations

PDP-6: Instruction Set



- Effective address calculation for every instruction:
- if X = 0: E := Y
- else: E := (X) + Y
- if I: fetch (E) and repeat (endless loop possible!)
- Two operands: AC and E

PDP-6: Full word move

- MOVE: move word
- MOVS: move word, swap halves
- MOVM: move word magnitude
- MOVN: move word negated
- modes:
- -: (E) $\rightarrow$ (AC)
- I(mmediate): 0,E $\rightarrow$ (AC)
- M(emory): (AC) $\rightarrow$ (E)
- S(elf): (E) $\rightarrow$ (E)/(AC)

PDP-6: Half word move

- move left/right half to left/right half and …
- -: don't modify other half
- Z: set other half to zero
- O: set other half to one
- E: set other half to sign
- modes: -, I, M, S
- HLLO: greetings!

PDP-6: Fixed point arithmetic

- ADD: add
- SUB: subtract
- IMUL: multiply (one word product)
- MUL: multiply (two word product)
- IDIV: divide (one word dividend)
- DIV: divide (two word dividend)
- modes:
- -, I, M, B(oth)

PDP-6: Floating point arithmetic

- FAD: add
- FSB: subtract
- FMP: multiply
- FDV: divide
- (R)ound
- modes:
- -, I, M, B(oth)
- FSC: scale by power of two

PDP-6: Arithmetic comparison

- CAI, CAM: compare AC and Immediate/Memory. skip on condition
- JUMP, AOJ, SOJ: (add/subtract one to/from AC and) jump on condition
- SKIP, AOS, SOS: (add/subtract one to/from (E) and) skip on condition
- conditions (always signed!):
- -: never
- A: always
- E, N, L, LE, G, GE: what you expect

PDP-6: Boolean

- all operations can complement the output (different mnemonics)
- SETZ/SETO: set to zero/one
- AND/ORCB: and/or compl. both
- ANDCA/ORCM: and compl. AC/or compl. mem
- SETM/SETCM: set to mem/compl. mem
- ANDCM/ORCA: and compl. memory/or compl. AC
- SETA/SETCA: set to AC/compl. AC
- XOR/EQV: exclusive or/equivalent
- IOR/ANDCB: inclusive or/and compl. both
- modes: -, I, M, B
- standardized in Common Lisp!!

PDP-6: Testing and modification

- Test AC with right/left/direct/direct swapped word and…
- N: do nothing
- Z: zero bits
- C: complement bits
- O: set bits (to one)
- …skip if: -, A, E, N
- TRON: 666

PDP-6: Byte operations

- IBP: increment byte pointer
- LDB: load byte
- DPB: deposit byte
- ILDB, IDPB: increment and load/deposit byte

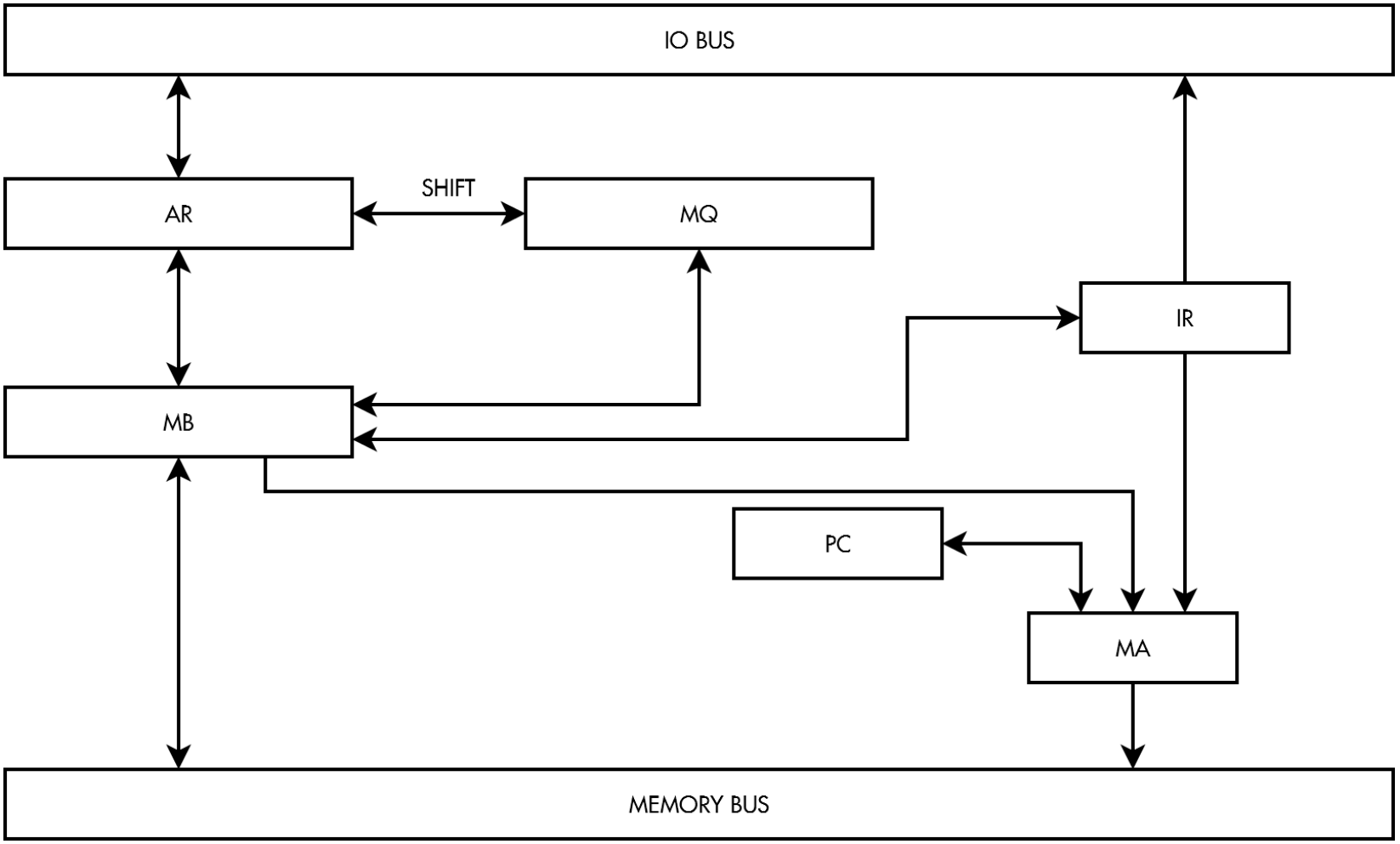
PDP-6: Misc instructions

- ASH, LSH, ROT: shift, rotate. -C: two ACs combined
- EXCH: exchange AC and (E)
- BLT: block transfer
- AOBJP/N: add one to both halves and jump if positive/negative
- JRST: jump and reset
- JFCL: jump and clear flags
- XCT: execute
- PUSH/POP: push/pop word
- PUSHJ/POPJ: push/pop and jump (stack based calls)
- JSR, JSP, JSA, JRA: other subroutine jumps

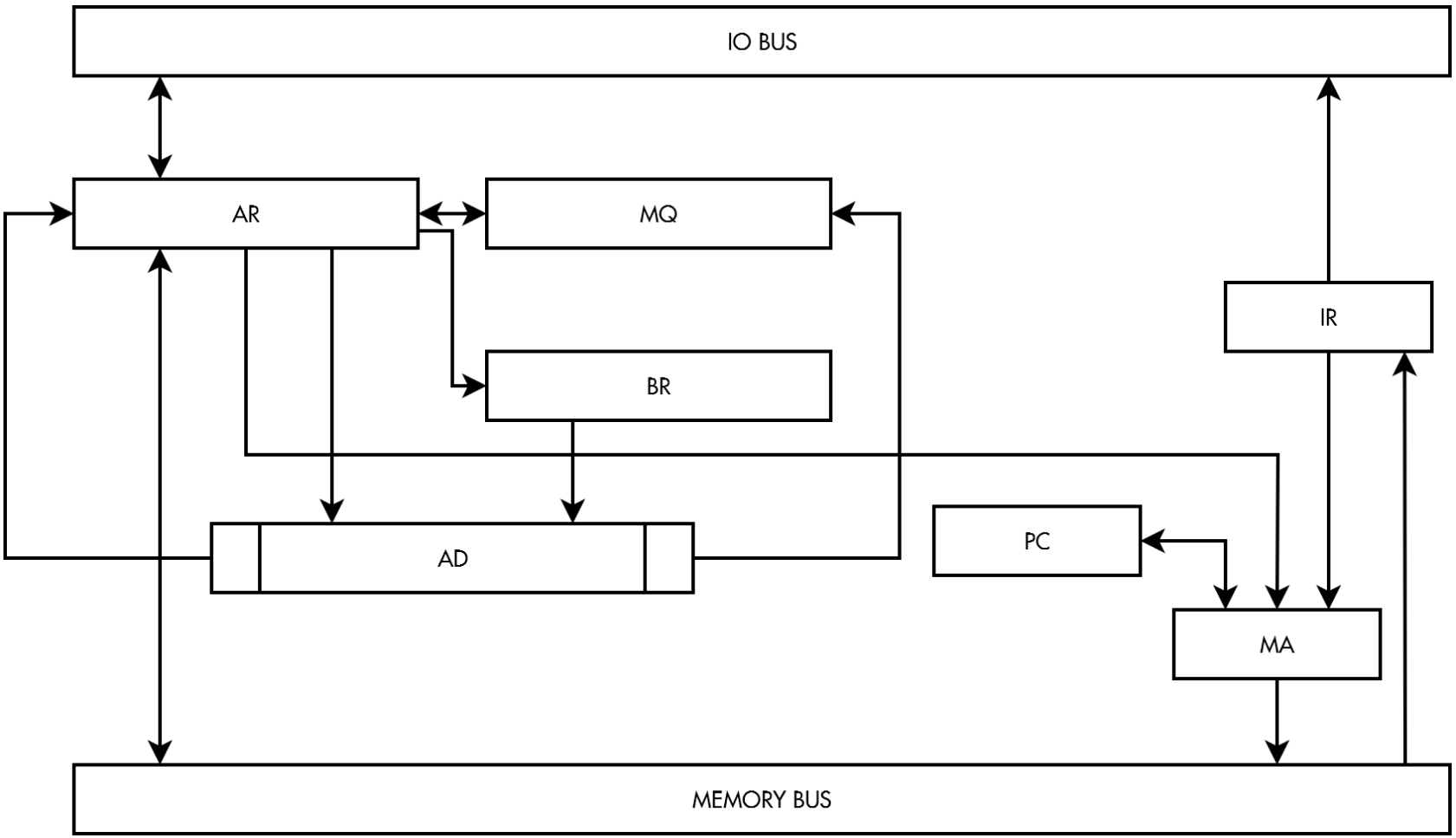
PDP-6: Microarchitecture

- PDP-6 and KA10 are asynchronous:
  no single central clock
- instead: chain of delay elements
- main cycles:
- K(ey)
- M(emory) C(ontrol)
- for each instruction:
- I(nstruction fetch)
- A(ddress computation)
- F(etch operands)
- E(xecute)
- S(tore result)
- various timing chains for non-trivial instructions

PDP-6: Simplified block diagram



PDP-10: Simplified block diagram (KA10)



PDP-6: Microarchitecture

Hardware subroutines:

```
IT1:
  MA <- PC
  IF1A <- 1
  trigger MC READ RQ PULSE
IT1A:
  IF1A <- 0
  (instruction is now in MB)
  ...
```

```
MC READ RQ PULSE:
  ...
  talk to memory
    to read word at MA
    into MB
  ...
MC RST1:
  if IF1A -> trigger IT1A
```

Used for memory access, addition/subtraction and all kinds of non-trivial instructions

Bootstrapping the PDP-6

- while the PDP-6 was being built, an assembler running on the PDP-4 was used to write code
- MIT people wrote LISP SUBRs on a blackboard and tested them on the prototype at DEC
- at MIT, no text editor running on the PDP-6 at first. Written over the weekend when DEC came to take away the PDP-1: TECO
- generally DEC and MIT people worked closely together

MIT software library

- TECO: text editor
- MIDAS: assembler
- DDT: debugger
- MACDMP: DECtape loader/dumper
- MACLISP
- later ITS ($\rightarrow$ Lars' talk)
- MACHACK VI: chess
- display hacks
- music
- various AI stuff

DEC software library

- DECDMP: DECtape loader/dumper
- MONITOR, later TOPS-10
- EDITOR
- MACRO assembler
- LOADER
- PIP: peripheral interchange program
- TECO and DDT from MIT
- FORTRAN II and IV
- …

PDP-6: A failure (?)

- PDP-6 was built only 23 times
- hard to maintain and very flakey
- financial disaster for DEC
- the architecture was loved and had potential
- DEC engineers start work on the PDP-10: same architecture but implemented better

PDP-10: A success

- Ken Olsen does not want another big computer and is tricked by making parts of the machine optional
- PDP-10 is a success
- DEC sold four generations: KA10, KI10, KL10, KS10
- killed off in the early 80s in favour of the VAX
- other companies make and sell their own PDP-10 implementations
- as of today, the only PDP architecture that is not legacy! lives on in XKL network gear