Processor Design

CPU → exe seq of instr in mem
- seq – instr cycle
- 2 parts

Fetch – obtain from main mem.

Execute – decode instr, fetch operand, performs operation (opcode)

CPU cycle time – time for shortest well defined micro-op

Clock rate –
- to supervise via control lines
- infrequent events handled by i-terrupts

[Dia. Overview of CPU]

as fast as possible; no. of components relatively small

- CPU contains registers
- registers faster

Prog Execution

① Transfer operands from MM to CPU-Reg.
② Compute
③ Transfer CPU to MM

I/O device communication with I/O ports (addressable registers)

1 Memory Mapped I/O – No IO instr.
- I/O data transfers implemented by memory mapped instruction (rights for bus)
- share same address space

2 I/O mapped or port-addressed I/O –
have IO instr. distinct from memory-referencing instructions

Accumulator – CPU register, store input and output operand in execution of most instruction

Polish Notation

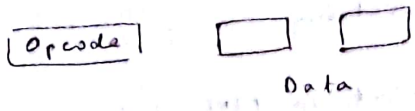
- Postfix
- Prefix

Adv - no parentheses

Opcode

Fixed length field of k bits - 2^k instructions / operations

- addressing info can be incl in opcode
- no. of reg small



- Some prg occupy much storage
- shortest format for frequently used instructions ...

Instruction Types

- one/2 register transfers, sequence of instructions...
- Due to complexity few successful attempts at HLPL
- semantic gap - compiler must bridge

Requirements —

1. Complete — Mach. lang. prog. in a reasonable amt of memory space
2. Efficient — Freq. reqd functions performed rapidly
3. Regular — contains expected opcodes and addressing modes
4. Compatible — orthogonal wrt addressing modes
   — reduce h/w, s/w costs instr reqd to be compatible — existing machines
5. Completeness — evaluated in finite no. of steps by Turing ...

5 categories of instr:

1. Logical
2. Prg Control — branch JMP
3. Arithmetic
4. Data-transfer
5. I/O

RISC v/s CISC

- no general agreement what constitutes appropriate size of instr ...
- early computers - small simple instr.
- h/w cheaper → instr sets complex
- contains hard to prog instr
- reduce semantic gap
- increase complexity

11-08-2018

Reduced instr set computer

1. Few instr. types + addr modes
2. Fixed and easily decoded instr
3. Fast single cycle instr cycle
4. Hardwired
5. Mem. access limited to load / store
6. Use of compilers to ~~obj~~ optimize object code performance

- Interval closely related
- Efficient- compilation requires architects and compiler writers to cooperate closely in design process.
  (optimization req^d)

- larger than usual no. of registers
- more code than CISC
- CISCs outperform RISCs in floating pt calculations

Implementation

register level logic design...

- balance cost with speed
- common to be stored
- reusability of algos.

} choose algos that allow circuits used by dill in

2 conditional ~~skip on~~ branch instructions SZA (skip on zero)
SNA (skip on non ze
accumulator

[Types - Adder, mult, ...]