Control Data 6600 Design Goal

- Beat IBM 7090 at number crunching
- Do it using 100 nsec gates (1963, discretes)

Key Idea

- Cpu as refrigerator, permitting
  - high drive currents for gates and memory cores, causing short switching times
  - dense packaging leading to short wires

CPU approach

- Build many specialized ALUs (functional units) so that as many as
  - 1 ADD
  - 2 MPY
  - 1 DIV
  - 1 LONG ADD

CPU approach

- 1 SHIFT
- 1 BOOLEAN
- 2 INCREMENTS
- 1 BRANCH

- can occur simultaneously

CPU approach

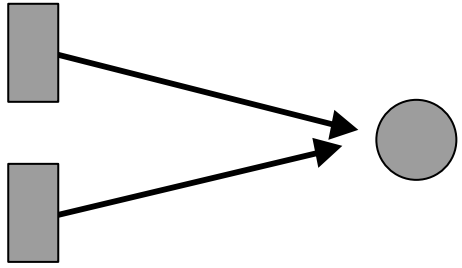
- Use short instructions
  - (4 per 60-bit word), quadrupling the fetch rate
- keep 10 instructions in a fast instr store
- offload small tasks
  - running OS
  - I/O
- to the PPUs

CPU approach

- Functional unit scheduling is programmer-transparent
- from 1 to 10 may be simultaneously active

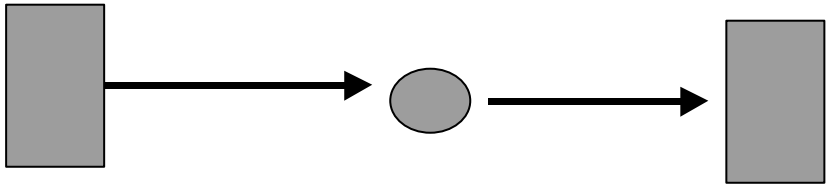
Scheduling rules

- Issue instructions until
  - no FU is free, or
  - some register would get 2 results



Scheduling rules

- Execute instructions until
  - an input depends on a non-existent output

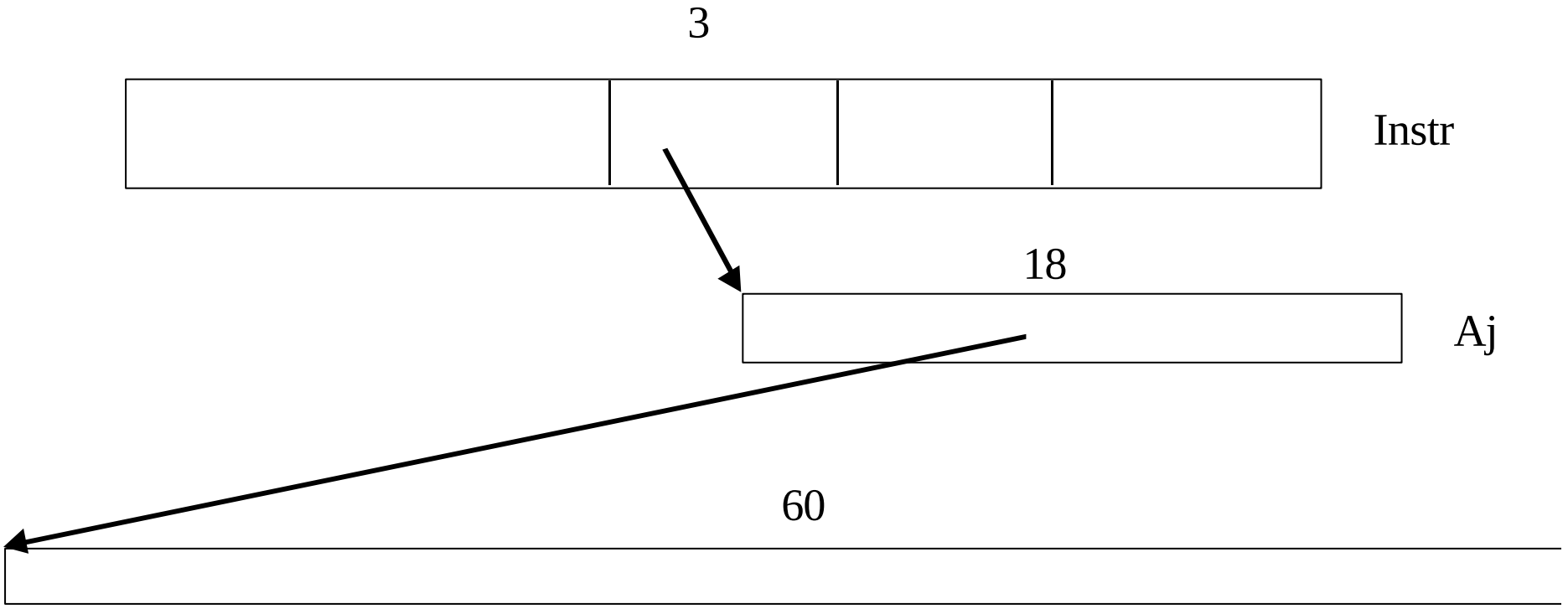


Instruction design

- Want 15-bit instructions
- need to include
  - op src1 src2 dest'n
- Mp addresses are 18 bits

Instruction design

- Store pointers to Mp addresses in instrs



Instruction design

- There are 8 pointer registers Aj
- for each there is a data register Xj
- when address -> Aj then
  - c(address) -> Xj 0 < j < 6
  - c(Xj) -> address 5 < j < 8

    - AUTOMATICALLY!

Example

SA1 A2

does

c(A2) -> A1 /* the address
c(c(A2) -> c(A1) /* the memory word

CDC 6600

There are also 8 index registers

B0 -B7, c(B0) = 0

C(A0) = 0 too

A program:

Add VEC1 to VEC2 component-wise, putting the vector sum in VEC3

```
ADD:  SB1    B1 - B1          ;      ZERO B1
      SB2    B1 + LENGTH;            LENGTH OF VECTORS
LOOP: SA1    B1 + VEC1;              A1 <- VEC1 + C(B1)
                              ;      X1 <- C(VEC1 + C(B1))
                              ;      I.e. VEC1j
      SA2    B1 + VEC2;              SAME DEAL FOR VEC2
      IX6    X1 + X2          ;      X6 <- C(X1) + C(X2) =
                              ;      JTH COMPONENT OF VEC1
                              ;      PLUS
                                     JTH COMPONENT OF VEC2
      SA6    B1 + VEC3        ;      A6 <- VEC3 + C(B1)
                              ;      C(A6) <- C(X6) =
                              ;      JTH COMPONENT OF SUM
      SB1    B1 + 1           ;      BUMP INDEX REG
      LT     B1,B2, LOOP      ;      END TEST
```

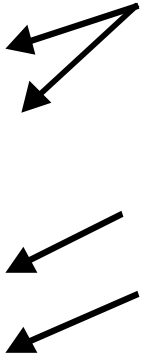
PARALLELISM ANALYSIS

THE RULES:
ISSUE UNTIL NO FU FREE OR SOME REG GETS 2 RESULTS

EXECUTE UNTIL A NECESSARY INPUT IS NOT YET AVAILABLE

PARALLELISM ANALYSIS

Box used	instr	clock times issued	started	finished
INC1	SB1(B1,B1)	1	1	2
INC2	SB2(B1,con)	1	2	3
INC1	SA1(B1, K)	2	2	3
INC2	SA2(B1,K0	3	3	4
ADD1	IX6(X1,X2)	1	4	5
INC1	SA6(B1,K)	3	5	6
INC2	SB1(B1,K)	4	5	6
INC1	LT(B1,B2)	6	7	7



WAIT FOR X2!
WAIT FOR X6!

PARALLELISM ANALYSIS

SAVES ONE CYCLE

HOW TO IMPROVE?