Multiplexed Information

And

Computing Service

Multics

as a

Software Factory

<u>Topic of Session</u>:

Use of Multics for the

<u>Maintenance</u>

and

<u>Extension</u>

of

Multics,

PL/I and

all user software

completely <u>on-line</u> with minimal

system interruption.

Overview

1. Introduction

History

	1965:	MIT/GE/BTL Joint Development Project
Fall	1965:	FJCC Multics Papers
	1967:	EPL Available; New PL/I begun
Spring	1968:	"One" user system--virtual memory credibility
Fall	1968:	"Five" user system
Fall	1969:	"30" user system--CTSS-like response (bad). New PL/I Multics "public"
Fall	1970:	40 load-unit system--better than CTSS response

Currently Underway:

New Features

- Version II PL/I
- APL
- GECOS environment
- Absentee batch processing
- etc.

Improving User Interfaces

- Better error messages
- Simplified commands
- etc.

Improving Capacity and Performance

- Expandable configuration
- Performance monitoring and analysis
- Software improvements
- Hardware improvements

Examples of Significant Software Development Achievements:

1. PL/I

   Began: 4Q67

   First Release: 4Q69

   4 Programmers

2. File System Redo

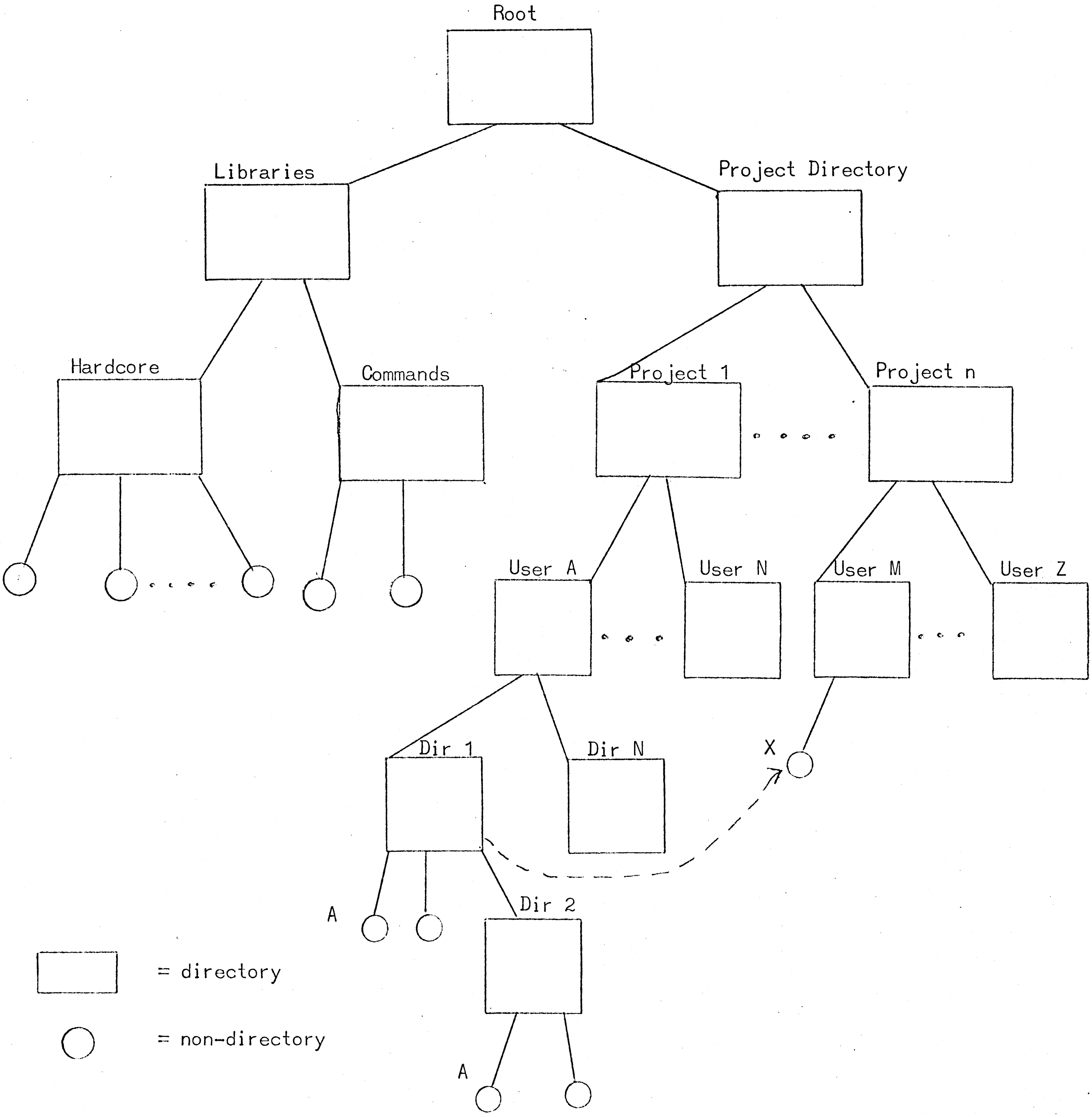
   ~ 50 modules "Opened Up"

   ~ 4-5 months

   2 Programmers

2. File System

Multics Directory Hierarchy



Multics Directory Hierarchy:

- System directories and files treated same as user directories and files

- Pathnames uniquely identify files

  root>project_directory>project1>userA>dir1>A

- Working Directory (Abbreviation)

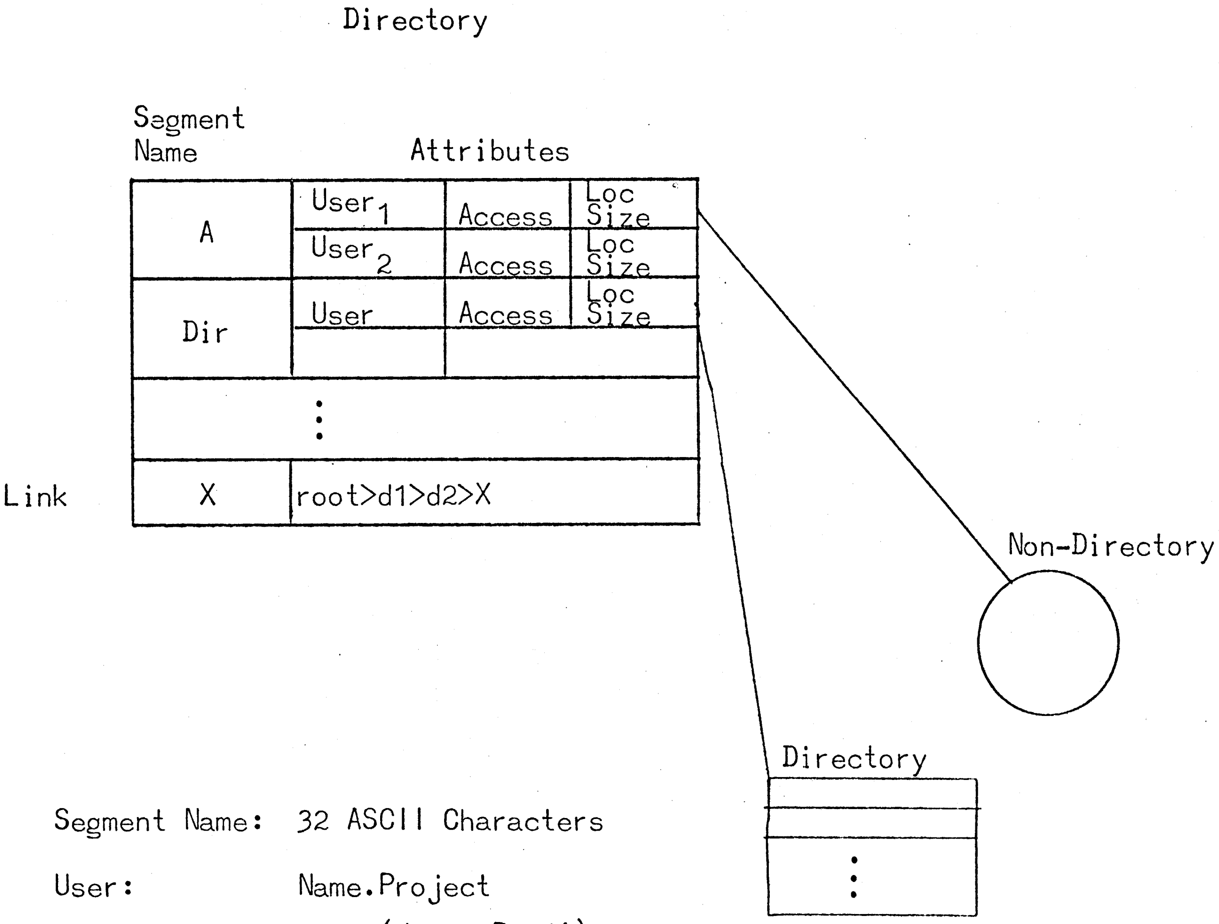
  cwd root>project_directory>project1>userA

  A

- Links (Indirect Addresses)

  X ("Indirect Pointer" to file in another directory)

Directory Structure and Segment Attributes



Segment Name:	32 ASCII Characters
User:	Name.Project (Jones.ProjA)
Access:	REWA - Non-Directory - Directory

File System Features

- Access Control
- Quota Control
- Source Code, Listings, Documents, Object Code, Data treated uniformly.
- Backup/Retrieval
- Commands to Manipulate Segments and Attributes:
  - List Directory Contents
  - Status of Single Segment
  - List and Set Access Control Info
  - Create and Delete Directory
  - Create and Delete Non-Directory
  - Rename Segments; Add Extra Names
  - Manipulate Quotas
  - etc.

3. Program and Document Preparation

Text Editors

(QED and EDM)

1. Interactive

2. General Purpose

3. Line Number or Context Driven

4. EDM is easy to learn

5. QED is more powerful

Examples of EDM
Commands

change

```
c5/abc/xyz/
```

delete

```
d10
```

find

```
f this is
```

locate

```
l reference
```

File Printing

1. Compiler listings and command outputs are files.

2. Files are printed:

   on-line (by print)
   off-line (by delayed print)

3. Delayed Print Features:

   a. three priority queues

   b. option to delete file

   c. identification option

4. RUNOFF command creates "type set" documentation.

4. Programming Languages

PL/I

- Standard Multics language
- Designed for system programmers
- Efficient object code
- Nearly full ANSI language
- On-line or off-line compilation

ALM

- 645 assembler
- Not intended for general use
- <5% of system written in ALM

APL (Iverson's not GE's)

- Interactive language
- Dynamic attribute assignment
- Compatible with IBM implementation
- Implemented in PL/I

FORTRAN

- Compatible with PL/I and ALM
- Superset of ANSI FORTRAN
- Version II compiler will use PL/I code generator

BASIC, LISP, SNOBOL, etc.

- Student projects

-Time Enviro

[illegible]time Environment

Virtual Memory

- procedure segments

  pure procedures

  automatic [illegible]

  static

  "data segments

- built dynamically
- very large [illegible]
- access

  to everything

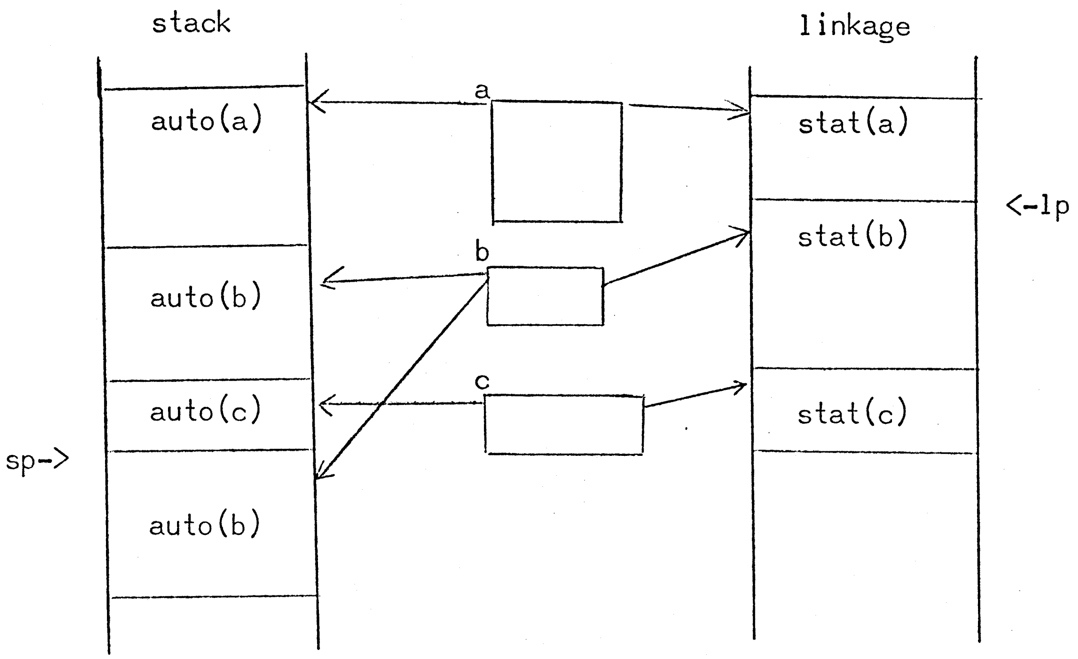
  control [illegible]

Other resources

- typewriter
- daemon [illegible]
- peripherals
- [illegible]

Procedure Environment

<u>a</u> calls <u>b</u> calls <u>c</u> calls <u>b</u>



standard call mechanism

standard data types

"binding"

- dynamic linker
- binder

6. Debugging Aids

Debugging Environment

1. Symbolic debugger

2. Dynamic trace

3. Quit mechanism

4. Simple I/O routines

5. Segmented "address space"

Debug

1. Symbolic

   - requests use source language
   - uses compiler produced symbol table

2. Interactive

   - requests given at run=time
   - no recompilation
   - concise system programmer oriented requests

3. Capabilities

   - inspect data or code
   - modify data
   - trace stack
   - control execution
   - machine language oriented features
   - escape to command processor

Example of use of debug

set break point

```
    /calc/read-line<
```

call procedure with arguments

```
    ||calc 37 41
```

breakpoint reenters debug

```
    Break 1 in calc
```

print data

```
    i

    21

    p->item.a.b(3)

    "1101"b
```

print lines of source program

```
    &a60,2

    60      x=q->a.b+7;
    61      call z(x,y);
```

Set new break point

```
    61 <
```

Continue execution

```
    c
```

Trace

1. Dynamic trace

2. Inserts procedure between called and calling procedure

3. Insert/remove with no recompilation

4. User can supply procedure

5. Standard procedure available

   - traces call, prints argument list
   - computes time spent

Quit

Interrupts process execution

State of computation saved

Can restart computation

Can use process to execute other procedures

Permits sequence

```
->
    QUIT
    •••execute commands
    start
```

Similar mechanism used for errors

Far-out example of quit

program output

```
    i = 1

    i = 1

    i = 1

    i = 1

→   QUIT
```

enter debugger

```
    debug

    /prog/i

    37

    /prog/calc,s2

    calc:       call write ("i =  ",j);
                return;
```

use editor

```
    || edm prog.pl1

    Edit

    l calc:

    c/j/i/

    p

    calc:       call write ("i =  ",i);

    w

    q
```

compile program

```
    || pl1 prog table
```

leave debugger

```
    q
```

resume computation

```
    start

    i = 37
    i = 38
    i = 39
```

Software Instal

and Maintenance

Software Installation and Maintenance

- on-line capability
- library in Virtual Memory
- change while system running
- use standard commands

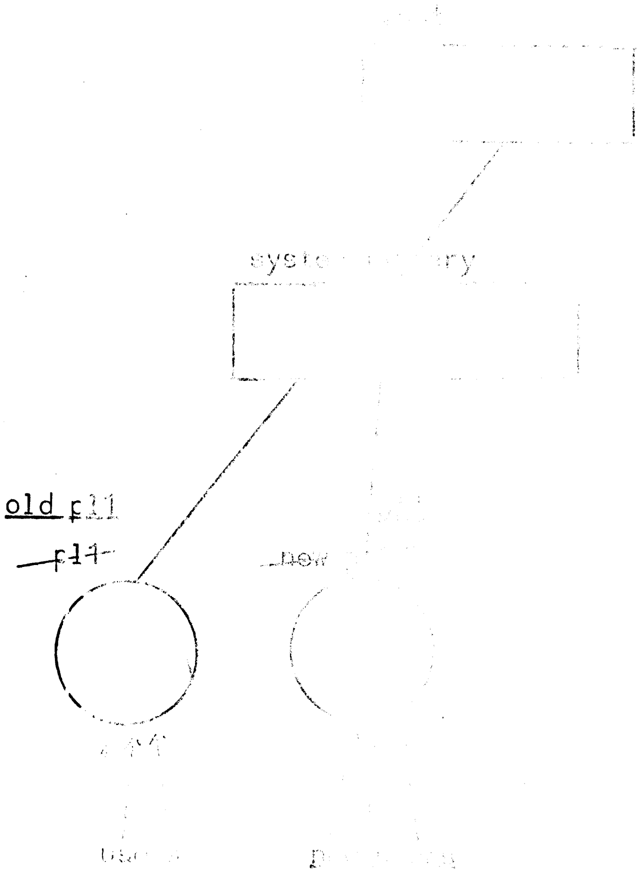
- Maintenance sequence

  source from library

  modify, checkout

  install

- easy to use others' programs


system library
old pl1
~~pl1~~
new

Software Quality

Ease of Use

1. Well designed user interface

   - took advantage of CTSS experience

   - meaningful error messages from compilers and system.

   - dynamic linking and file creation allow simple program execution.

2. Simple functional command language

   - system commands and user programs have equivalent interfaces.

   - commands are callable as programs:

   Example:

```
    pl1 alpha    =   call pl1("alpha");
    alpha beta   =   call alpha ("beta");
```

3. PL/I compiler options:

- source, symbols, map, list control listing output

- check - performs syntax and semantic error analysis

- brief, severity - control error messages

- optimize - removes redundant code on a per block basis.

- table - produces a run-time symbol table for symbolic debugging

4. Reliability

- Multics and PL/1 have been in use for more than a year.
- Heavy use by system developers and researchers.
- System and compiler are maintained by the original developers.
- Bug level is <u>near zero</u>.
- Failsoft design reduces system crashes due to hardware bugs.

5. RADC experience:

- 3 days to install first system

- runs 4 hours per day

- ran two months with:

  2 hardware crashes

  0 software crashes

Other Sources?

1. Batch processing

   - [illegible] (I/O [illegible]ansla[illegible]

   - [illegible] user (command [illegible]s; "canned" input)

   - [illegible] environment (GM[illegible], [illegible]TRAN, COBOL,
     A[illegible] "[illegible]rd" Interface,
     G[illegible]s).

2. Billing/record keeping for project management

   - Per project, per person, per shift

   - CPU time, console time

3. All system administration [illegible]

   - Adding/Deleting users

   - Changing passwords

   - Adding/Deleting projects

   - Modifying login privileges (e.g. guaranteed access)

   - [illegible]ling [illegible]

4. * ASCII [illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible] -300
- [illegible]
- [illegible] 5)

5. Read [illegible] write tape and ca[illegible]

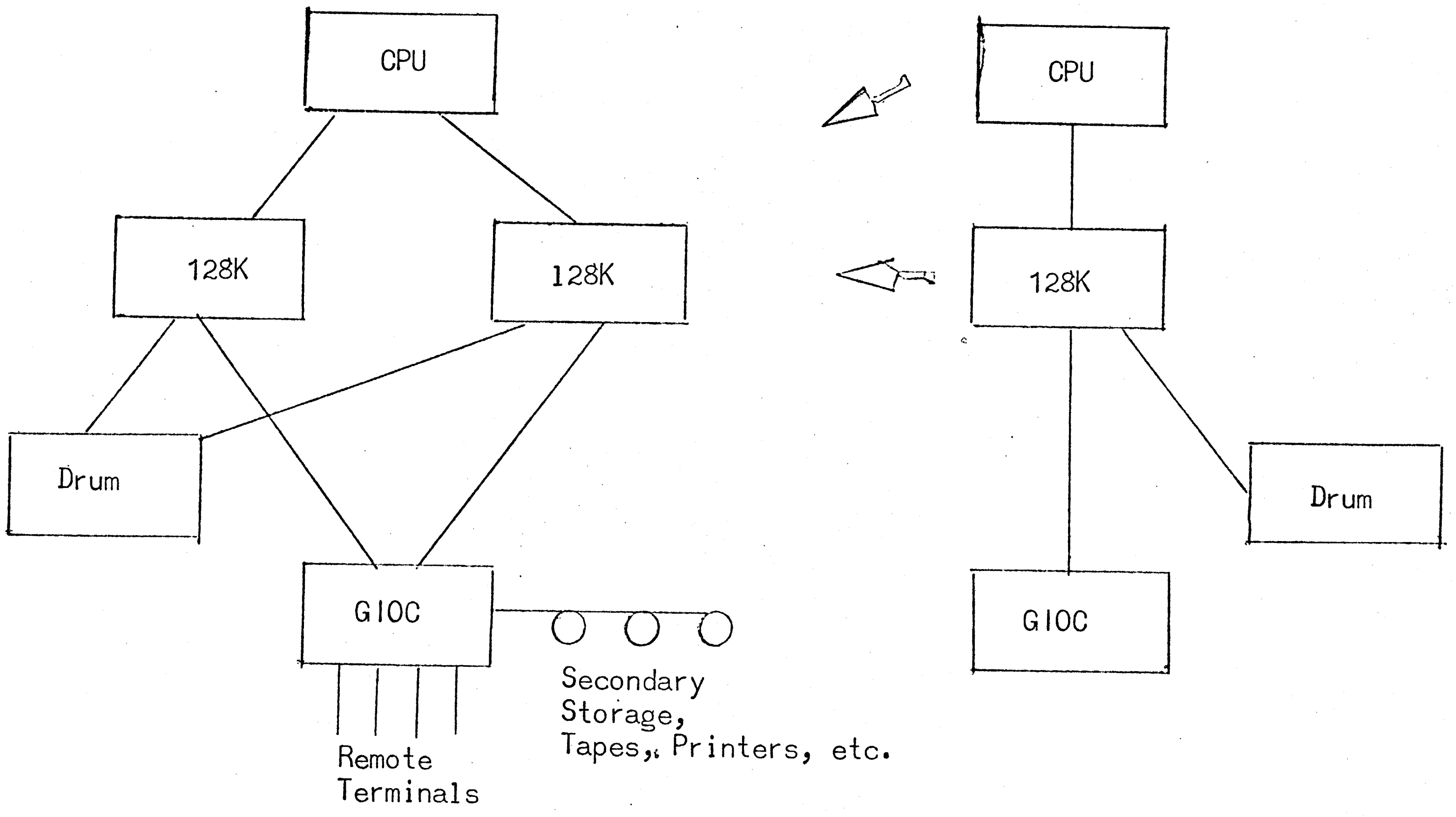
6. Ring [illegible] red [illegible] dom[illegible]

   - [illegible] on of system [illegible] mode
   - [illegible] refinement of [illegible] control [illegible] by segmenta[illegible] Proprietary/ [illegible] ted [illegible] data [illegible].
   - [illegible] simulate/ [illegible] [illegible] reign supervisors.

Configuration-[illegible]ity

and Performance

1. Configurations



- 256K + 1 CPU ⟶ 1M + n CPU's
- Dynamic Reconfiguration
  - CPU's
  - Memories
- Add Datanet-300 and GECOS-III with Time-Sharing can run on the configuration. (K5).

2. Capa[illegible]

>500 [illegible]

>90 [illegible]

~15 [illegible]

~1 2[illegible]

3. Perf[illegible]

Curre[illegible]

[illegible] ster programm[illegible] [illegible] mittent batch streams

[illegible] RTRAN [illegible] ([illegible] F AN compiler)

[illegible] ting users