Multics.

Charles Ahern
Scott Roffman

Topics

- What is Multics?
- Brief History
- Notable Features of Multics
- Influence on Other Systems
- Review
- Sources

MULTiplexed Information and Computing Service

- Multics is a timesharing OS begun in 1965 and used until 2000.
- Primary usage was with a mainframe and multiple terminals.
- CPUs, memory, I/O controllers, disk drives could be added or removed while the system is running

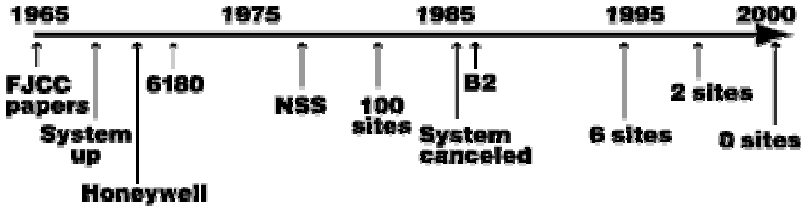
MULTiplexed Information and Computing Service

- Designed to run 24/7
- Changed the idea of the computer from being a tool for scientists to a reliable and powerful resource for a large number of people

"Multics keeps it up longer"

Brief History

- Joint project between MIT, Bell Labs, and GE
- Bell labs withdrew in 1969
- GE Sold its computer business to Honeywell in 1970 who sold Multics as a commercial product



Features

- High-level language implementation
- On-line reconfiguration
- Large virtual memory with segments, paging, and generalized addresses
- First hierarchical file system
- Dynamic linking and function call by name
- Shared memory multiprocessor
- Security and rings

Language Implementation

- Written in PL/I language
- In 1965 this was a new proposal by IBM
- Only a small part of the OS was written in assembly
- Writing an OS in a high-level language was a radical idea at the time

Virtual Memory

- Divided into as many as 2^{14} *segments*
- Each segment has as many as 2^{18} 36-bit *words*
- Each segment is a logical unit of information with attributes for length and access privilege

Types of Segments

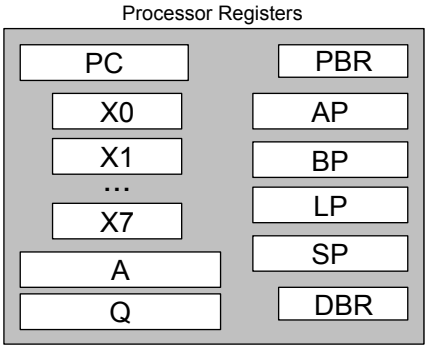
- 2 Main types of segments
  - Procedure
    - Intended to be accessed for an instruction fetch
    - Normally cannot write to a procedure segment
    - Reading may be prohibited if in use
  - Data
    - Contains no instructions
    - May or may not be write protected

Directory Structure

- Multics (at least in this era; pre-1981) does not speak of "opening" files. Multics supports a call to "initiate" a segment that maps onto an entire file.
- Hierarchical arrangement of directories that associates at least one symbolic name (perhaps many) with each segment.
- The term "file" and "segment" are often used interchangeably as a result of this one-to-one binding.

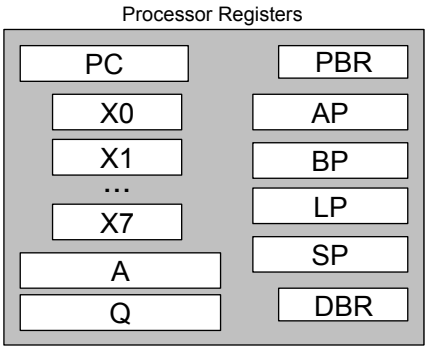
Standard Processor Registers

- The PC Is the Program Counter
- X0 through X7 are the Index Registers.
- A is the Accumulator register
- Q is the quotient register



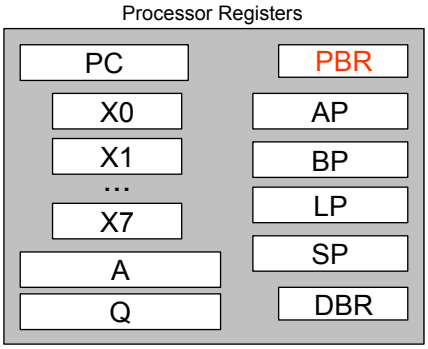
Base Pair Registers

- AP, BP, LP, and SP are the 4 base pointer registers
  - Argument Pointer
  - Base Pointer
  - Linkage Pointer
  - Stack Pointer
- They each hold a complete generalized address and are named according to their function in Multics.



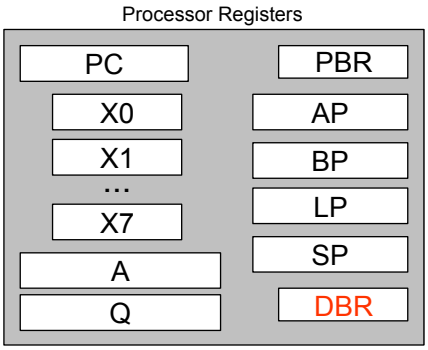
Other Registers

The PBR is the Procedure Base Register. It contains the segment number of the procedure in execution (think of it as your process's unique id).



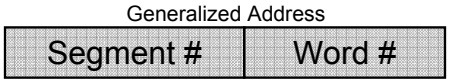
Other Registers

The DBR is the Descriptor Base Register. It points to the descriptor table for your process, which tells your process its security rights for each segment it is using and a pointer to each of them.

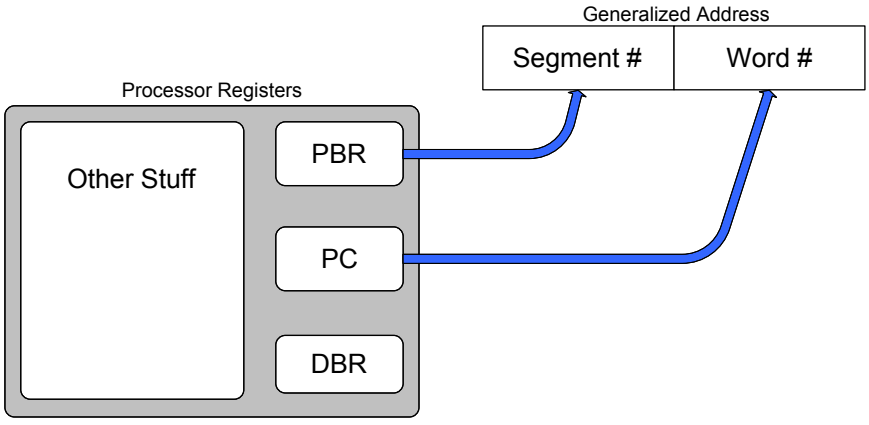


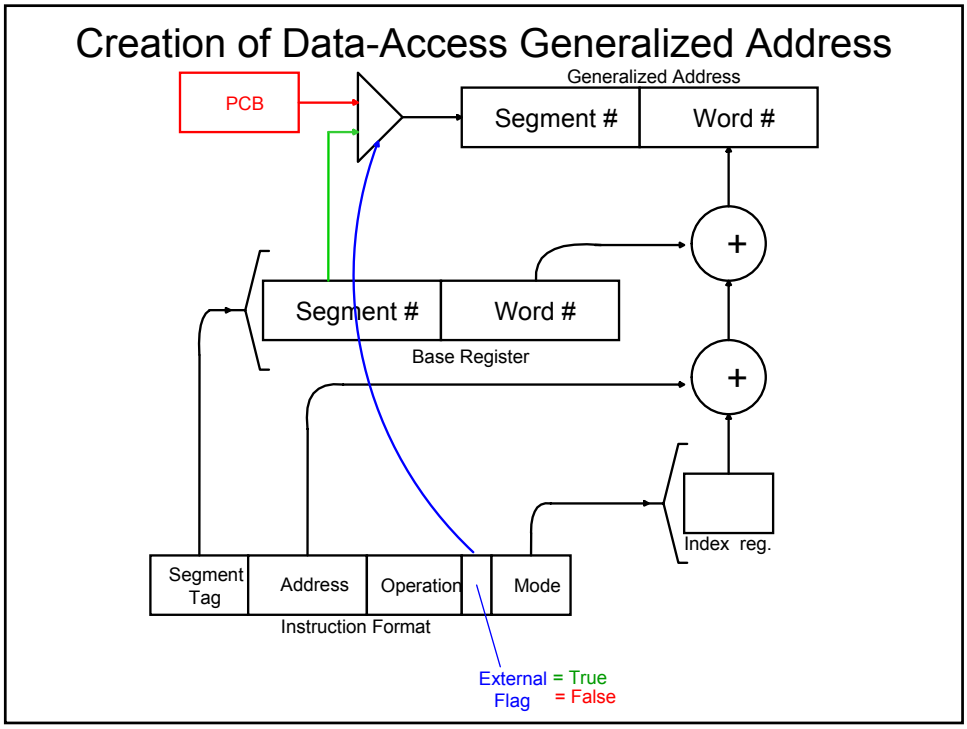
Addressing

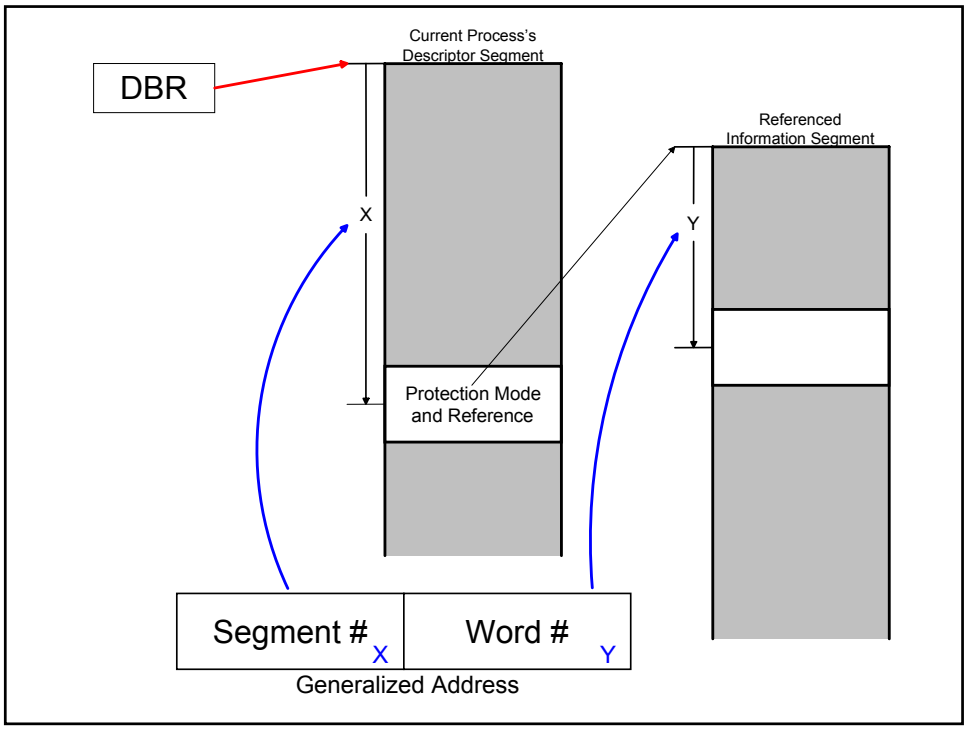
- Multics uses a "Generalized Address"
- It is calculated differently depending on if the CPU is attempting to read an instruction or data



Creation of the Instruction Fetch Generalized Address




Creation of Data-Access Generalized Address
Generalized Address
PCB
Segment #
Word #
+
Segment #
Word #
Base Register
+
Index reg.
Segment Tag
Address
Operation
Mode
Instruction Format
External Flag
= True
= False


Current Process's Descriptor Segment
DBR
Referenced Information Segment
X
Y
Protection Mode and Reference
Segment # X
Word # Y
Generalized Address

Generalized Address Benefits

- Reallocation of the processor to a new process requires little more than swapping out the DBR.
- The Generalized address allows access to a word without knowing it's physical location in memory

Inter-segment Linking

- Allows for the ability to share procedure and data information and the power to construct complex procedures by building on previous work.
- Location Independent Addresses are essential to performing these tasks.

Inter-segment Linking Requirements

1. Procedure segments must be pure.
   - Execution must not cause a single word of their own content to be modified.
2. It must be possible for a procedure to call a routine by its symbolic name.
   - Without prior arrangements.
3. Segments of procedures must be invariant to the recompilation of other segments.

Requirement 1

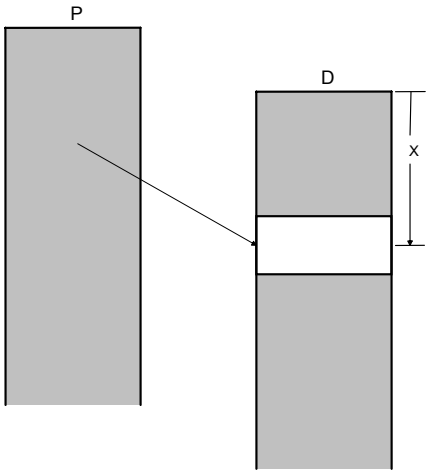
- Requires that a segment be callable even if no position in the descriptor segment of the process has been reserved for the segment.
- Making the segment *known* to the process is done by assigning a position in the descriptor segment (a segment number) when the process first makes reference to the segment, by its symbolic name.

Linkage Data

- Consider a procedure segment P that makes reference to a word at location x within data segment D.
- Ex. OPR <D> | [x]
  - The <> indicate that D is the reference name
  - The [] indicate that x is a symbolic address within an external segment

OPR <D> | [x]

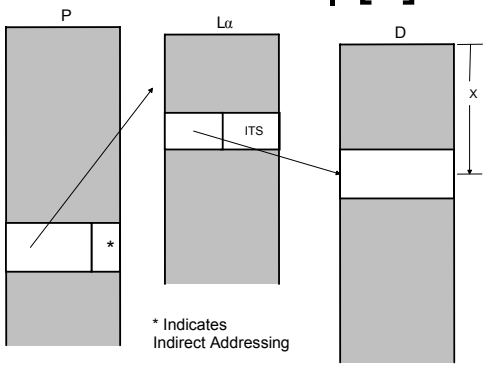
- Here we see the inter segment reference to x in D from procedure P.
- But to maintain the Third requirement we need segment p to be invariant to recompilation of D.



OPR <D> | [x]

- In order to adhere to requirement 3 and 2, we create a linkage section that contains all external references of procedure P.
- Each different process gets its own linkage section for each procedure it calls.
  - In our case there is an Lα, that would contain the reference to the actual location of x in D at the present time.

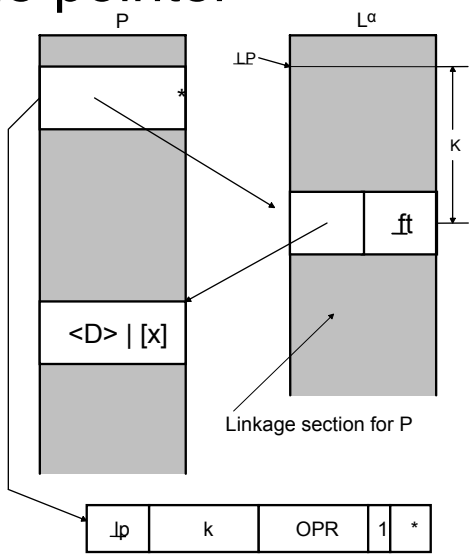
OPR <D> | [x]



- its stands for indirect to segment
- Before the link is established, it must verified, (lead to a trap, ft) and then search for the symbolic address <D> | [x]

Linkage pointer

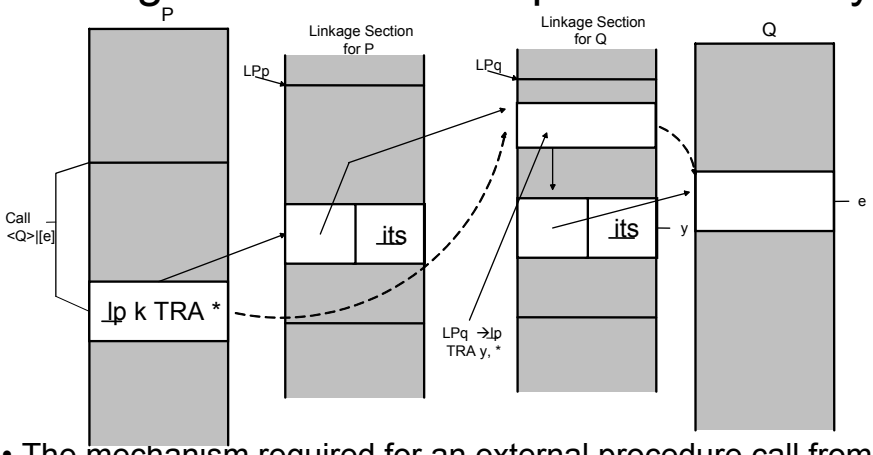
- The linkage pointer is a generalized address that resides in a dedicated base register (lp).
- The displacement k is determined by the coding of P and is invariant with respect to the process using P.



Procedure Call and Returns

- Conventions for four aspects of sub routine calling
  - Transmission of arguments
  - Arranging for return of control
  - Saving and restoring processor state
  - Allocating Private storage for called procedure
- The argument pointer (ap) (at procedure entry) contains the generalized address of the list of arguments for the called procedure

Linkage Mechanism for procedure Entry



• The mechanism required for an external procedure call from procedure P to segment Q at entry point e.

- The solid lines indicate the individual steps taken through indirect addressing.
- The dashed lines indicate resulting flow of control.

Executing call to external procedure

- The procedures base register and program counter are saved in the stack segment by the caller, pointer to by stack pointer (sp)
- Return from the called procedure can be effected by restoring the caller's machine conditions from the stack segment.

Unix and Multics

- Dennis Ritchie and Ken Thompson, creators of Unix, originally worked with Bell Labs on the Multics project
- When Bell Labs pulled out in 1969, Ken and Dennis began a the Unix Project
- The title “Unix” is a parody of Multics, “One of whatever Multics was many of.”

Conclusion

- High-level language implementation
- 24/7 “Computing Service”
- Security and rings
- Large virtual memory with segments
- Generalized Addresses
- Dynamic linking and function call by name

References

R. Daley, and J. Dennis, “Virtual Memory, Processes and Sharing in MULTICS” Communications of the ACM. Vol. II. Number 5. pp. 306-312. May, 1968.

A. Silberschatz , P. Galvin, and G. Gagne, "operating System Concepts" John Wiley & Sons, 7th Edition, 2005

Paul Green, “Multics Virtual Memory – Tutorial and Reflections” ftp://ftp.stratus.com/pub/vos/multics/pg/mvm.html

http://www.multicians.org/