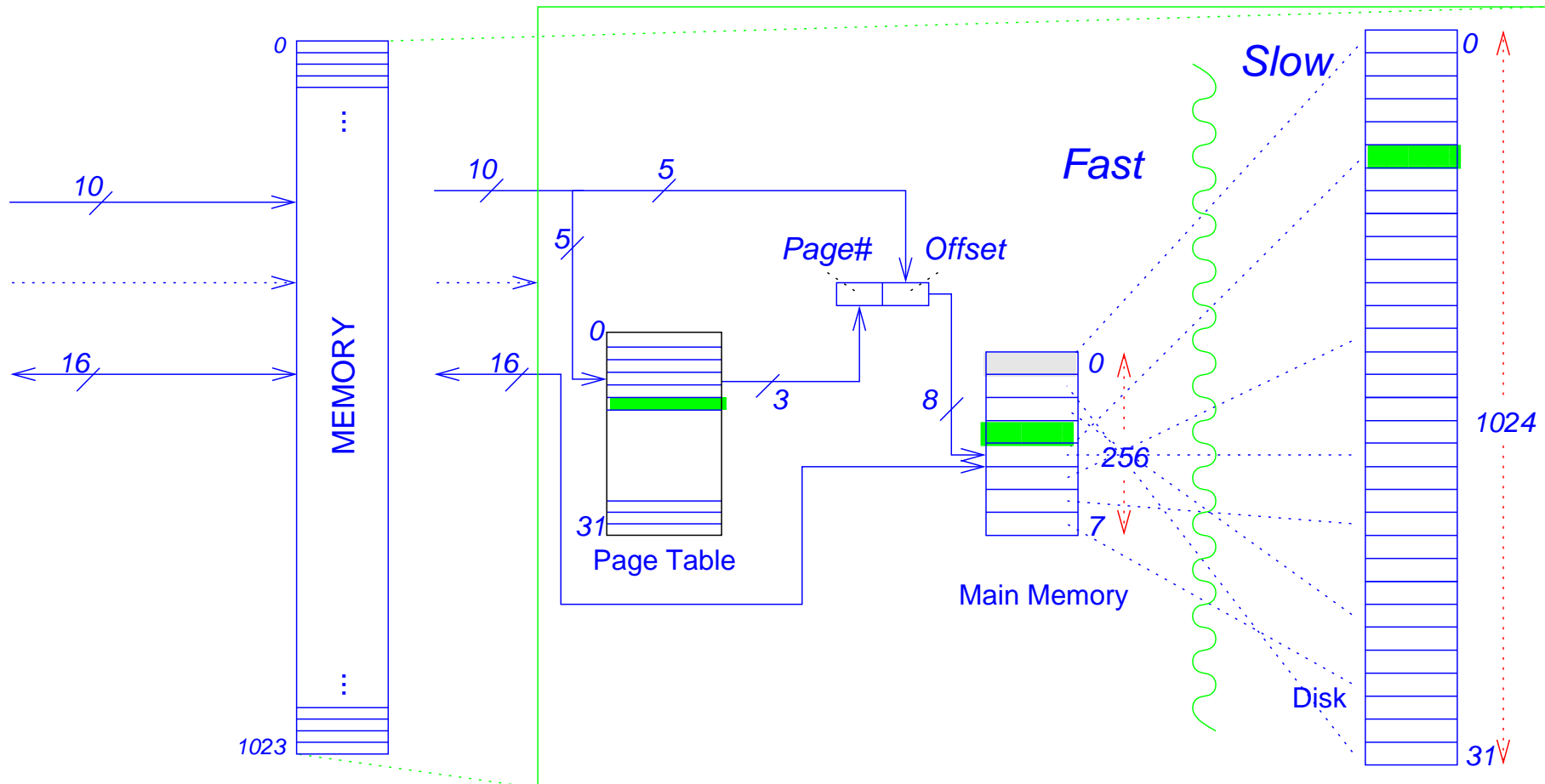


## Virtual Memory in PeANUt

- ref: [PeANUt Spec, sect 3]; additionally [O'H&Bryant, sect 10.1–10.7] or [Null&Lobur, sect 6.5]
- virtual memory implementation
  - page tables
- virtual memory in PeANUt
  - page replacement
  - paging behaviour
  - working sets
- other issues:
  - calling procedures inside procedures: example in appendix to lect P8
  - reminder: no labs next week – work on Assignment 2!



# The PeANUt Virtual Memory System



# Page Tables

- purpose: give information on the status of each of the pages of the virtual memory
  - is it present in main memory? Present bit
  - if so, in which page frame is it? Frame number field (3 bits)
  - if so, has data been written to it? Dirty bit

Other information	D	P	Frame
-------------------	---	---	-------

- where does this page table reside?
  - possibly in special hardware
  - in PeANUt: part of main memory, at VM addresses 0–31 (large: 1/8 of main memory!)

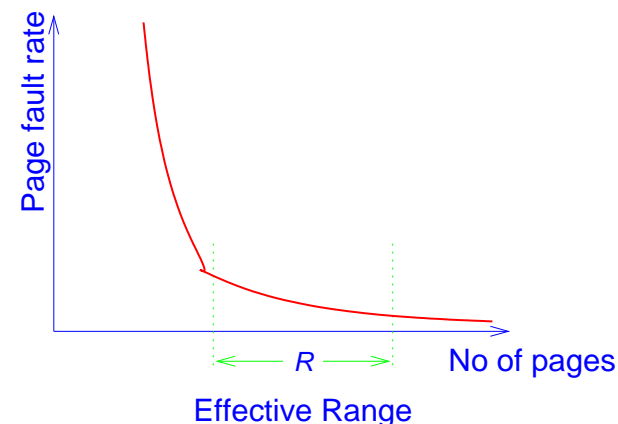






# Paging Behaviour

- how many pages? (or, how big are the pages?)
  - too few (too big) can result in un-needed data put in main memory, resulting in fragmentation, in turn causing page faults (when in 'steady-state')
  - too many results in huge page tables (and other large per-page overheads, including a large number of page faults initially, with a large number of disk accesses)

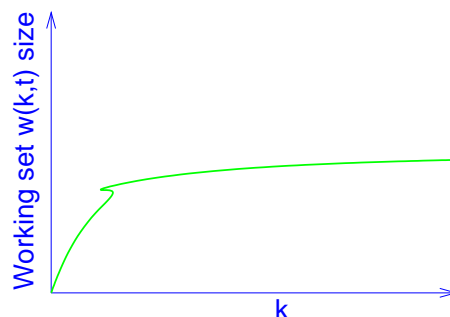


when program is in steady-state:

Assumption: page demand  $<$  available pages in main memory

## Working Set

- the working set function  $w(k, t)$  is defined as the set of pages accessed during the  $k$  most recent memory accesses, up until time  $t$



- as  $k$  increases, the size of the  $w(k, t)$  approaches a (constant) limit
  - this limit is the working set size at time  $t$
  - it is roughly independent of  $t$
- the working set size concept can be used to describe the memory needs of a program
  - if the main memory is not large enough to hold the working set, there will be excessive page faults (such behaviour is called thrashing)

is also an important concept in the memory hierarchy

## The Page Table in PeANUt

- is permanently located in page 0 (addresses **a0** to **a37**)
- it has 32 entries, each relating to one of the 32 VM pages
- 5 bits of a memory (VM) address are used to determine the page frame number, 5 bits for the offset within the page

- format of a page table entry:

Unused	Last used	Swap count	D	P	Frame number				
15	11	10	8	7	5	4	3	2	0

- page table fields:
  - Last used: reflects how recently the page has been accessed (a value of 0 indicates that this is the most recently accessed page)
  - Swap count: indicates how many page faults have occurred since this page was loaded into memory (indicating its age)
  - Dirty flag: a value of 1 indicates that data has been written to this page since it was swapped into memory
  - Present flag: a value of 1 indicates the page is present in main memory
  - Frame number: states which page frame (slot in main memory) a page is in (*if* the page is present)

## PeANUt Virtual Memory Traps

- **trap #11:** page fault (exception)

Is not user-initiated or user-definable. Can occur in the fetch of an instruction or the evaluation of its operand. Note: it causes *re-execution* of the faulting instruction.

- **trap #12:** swap page in

Move page number AC from secondary memory to main memory. The O/S will use page table entry AC (at Mem[AC]) to determine the destination page frame

- **trap #13:** swap page out

The O/S will move page number AC from main memory to secondary memory

- the handler routine for **trap #11** will use **trap #12** and **trap #13**, in accordance to which paging policy it uses

- example using the FIFO policy: vmfifo.ass

- Q: what if the handler routine for **trap #11** was swapped out? Similarly for the page tables? Can the routine use the stack (safely)?