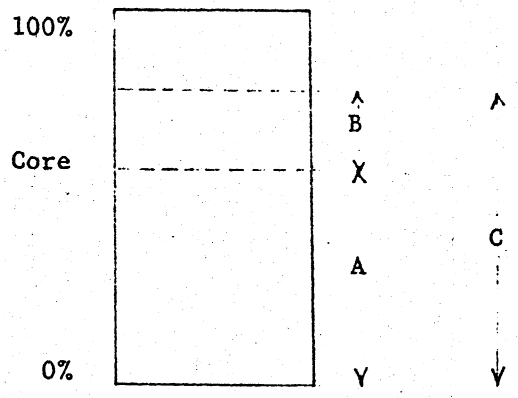
Thoughts on Paging

D. E. Joel 5/28/65

CORE FLUSHER

Function is to examine core usage, and if the percentage of core used has reached the threshold (system parameter), to effect the removal of the necessary number of pages to bring the core used down to the completely flushed level (system parameter).

Routine is invoked on a time basis. The time interval must be a system parameter.



Action: Amount of core used = C + Δ ($\Delta > 0$), pages removed to release core = B + Δ.

Amount of core used $<$ C, no action.

Page Usage Measurer

Function is to examine pages in core, and determine the rate of usage. This is done by looking at the use bit in the page table word, turning it off if on, and keeping a short history of the immediate past (say the last 10 looks). Pages which are marked as "HOLD IN CORE" are exempt (if this is helpful). Routine is invoked on a time basis. The time interval must be a System parameter.

This routine could be bypassed (to reduce overhead) when core usage is low - perhaps when usage is below the "completely flushed" level.

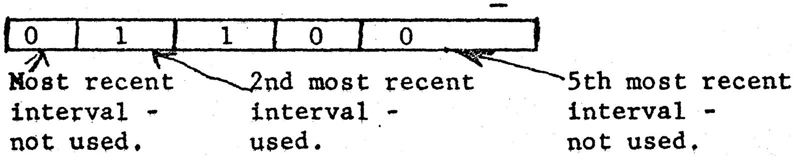
Requirements:

Table of pages in use. Each entry must contain:

(a) Pointer to page table word (24 bits)

(b) Recent history: possible method - N bits, most significant being most recent time interval, 1 indicating used, 0 indicating not used. This field could be used as a measure of the usage rate.

e.g. N = 5


0
1
1
0
0
Most recent interval - not used.
2nd most recent interval - used.
5th most recent interval - not used.

Page with history 01101 has higher usage rate.

Page Remover

Determines which pages to remove in order to account for the area $(\beta + \Delta)$ which the Core Flusher requests.

Considerations:

1. Page must be available to be removed i.e. not marked "HOLD IN CORE".
2. Usage rate should be relatively low.
3. Pure procedure is a good candidate - does not require writing out.

   Unchanged data or impure procedure is a good candidate.

The Technique of Paging

Each file (segment) has associated with it a table of physical locations (TPL). In this way a randomly dispersed file is described in a sequential manner. An entry in the TPL defines:

(i) A fixed amount of storage (-preferably a submultiple of 1024, Mod. 64)

(ii) The location in the physical domain at which this storage is located. Note that this pointer may not be the physical address, but may have a value such that it will yield the physical address when massaged by the appropriate algorithm.

Each segment in a users process will have a TPL. This table might beconveniently put into the users linkage segment.

In addition to the defined content above, each entry has a descriptor field. The entries which can occur are:

(a) Nothing filed.

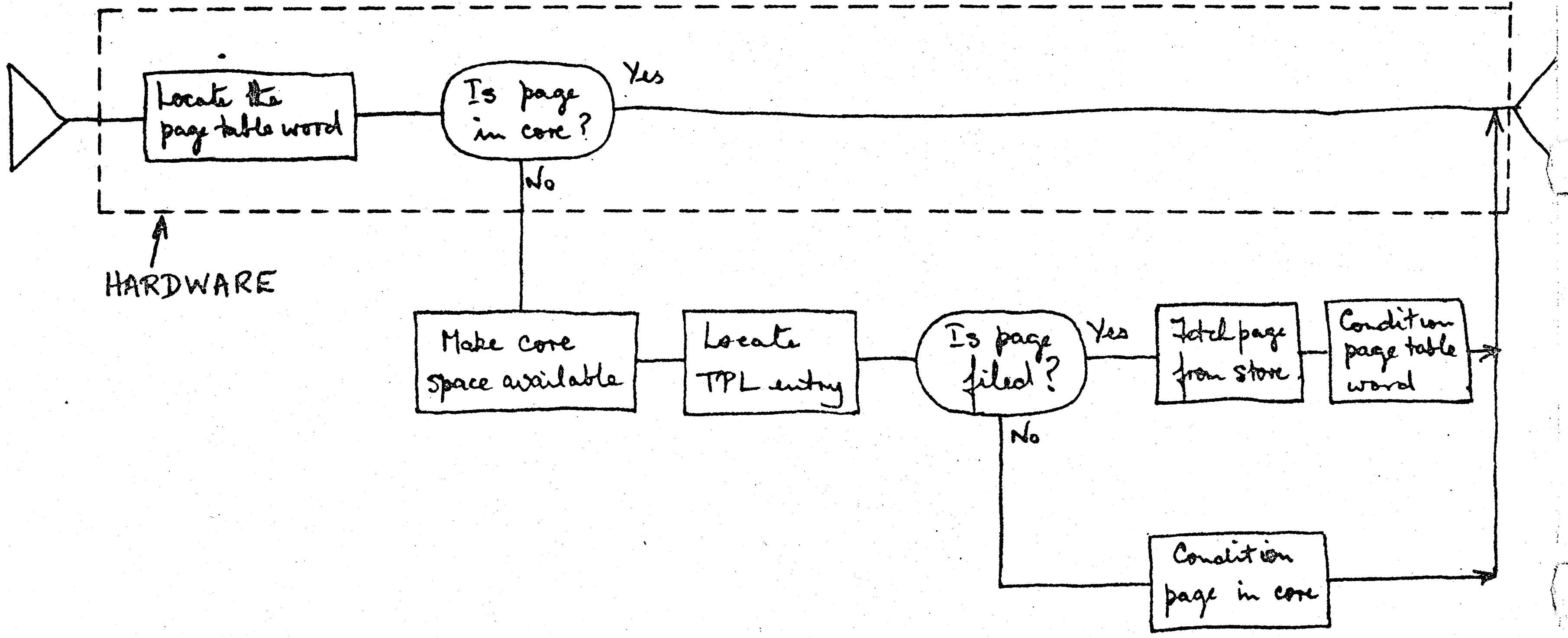
(b) Original file.

(c) Page file.

The attached flow charts show how the TPL is used in control of paging.

1. CORE ACCESS

An effective relative address has been calculated


Locate the page table word
Is page in core?
Yes
No
HARDWARE
Make core space available
Locate TPL entry
Is page filed?
Yes
No
Fetch page from store
Condition page table word
Condition page in core