PDP-1 COMPUTER
ELECTRICAL ENGINEERING DEPARTMENT
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
CAMBRIDGE, MASSACHUSETTS
02139

PDP-35

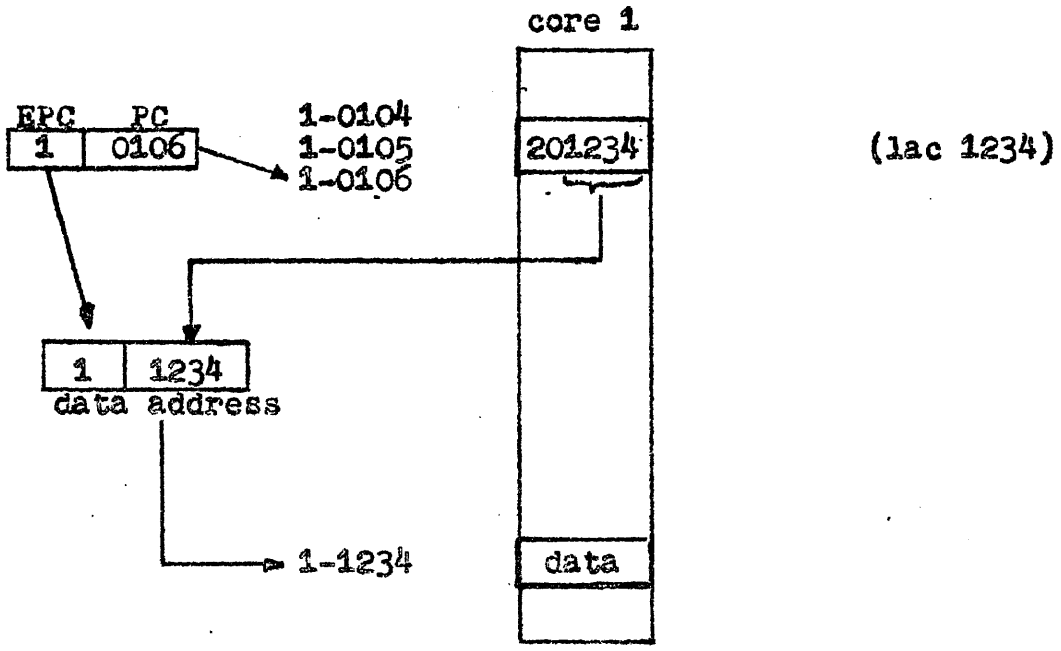
INSTRUCTION MANUAL

PART 2 -- ADDRESS CALCULATION

December 12, 1971

2.0 Address Calculation

A direct or unmodified data address is calculated by concatenating the Extended Program Counter (the EPC is bits 3-5 of the G register) and the address part of the instruction.



Addresses may also be deferred (indirected), indexed, or both (in either order). These operations are said to "modify" the address specified by the address part of the instruction.

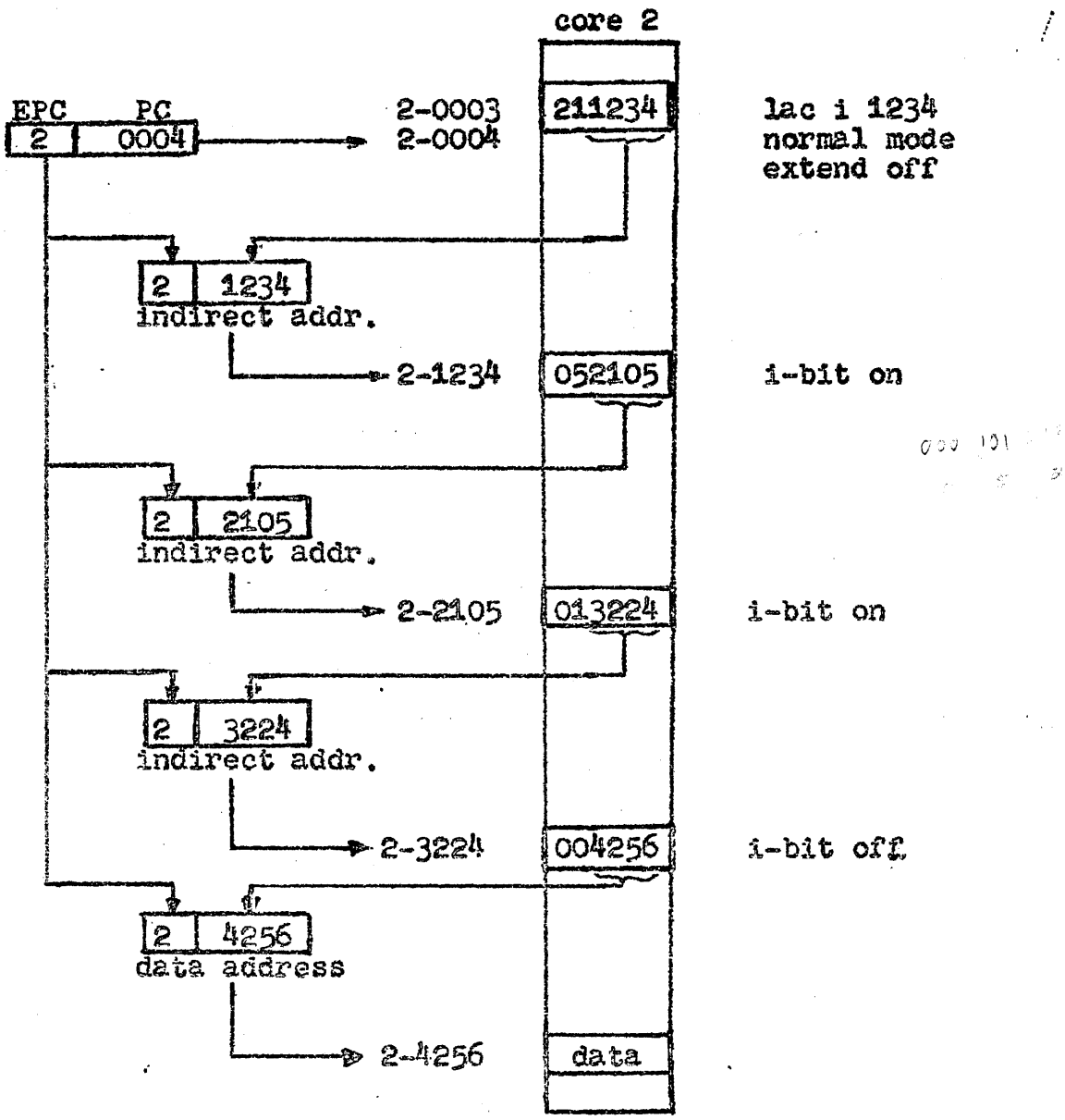
2.1 Indirect Addresses

Under certain conditions an instruction may specify that its address is indirect or deferred. This means that, instead of addressing the data directly, its address part gives the address of a location which contains a "pointer" to (address of) the data location. Each defer cycle takes one extra memory cycle.

Exception: alternated index mode, i-bit on

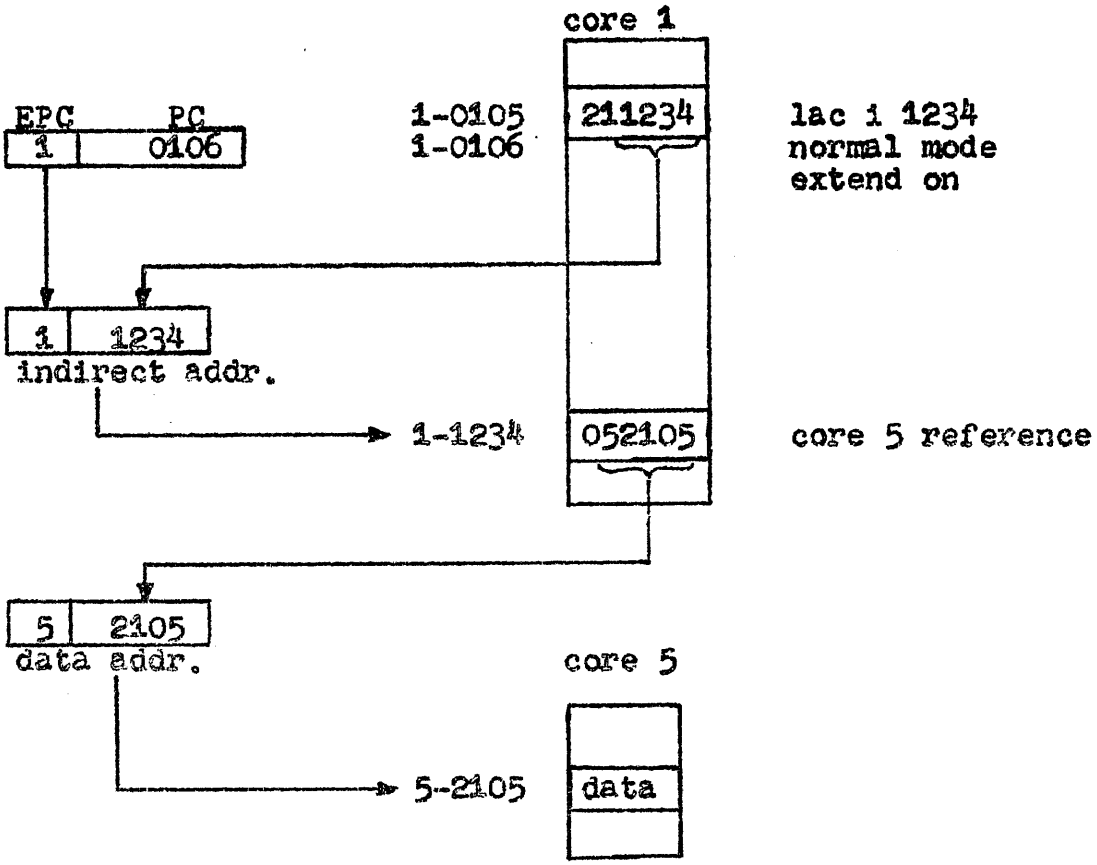
2.1.1 Indirect Addressing Out of Extend Mode

If the computer is not in extend mode (EXD is off), multiple level indirect addressing may be used, but the instruction, all defer cycles, and the final execute cycle will be in the same core memory module. After each defer cycle, the i-bit (bit 5) of the word read from memory is examined. If it is on, another defer cycle occurs at the address given by the address part of the word. If the i-bit is off, the indirect chain terminates and the address part is used as the actual data address.



2.1.2 Indirect Addresses in Extend Mode

With EXD on indirect addresses are single-level only. This deferred address is taken as a 15-bit quantity which may reference another core.



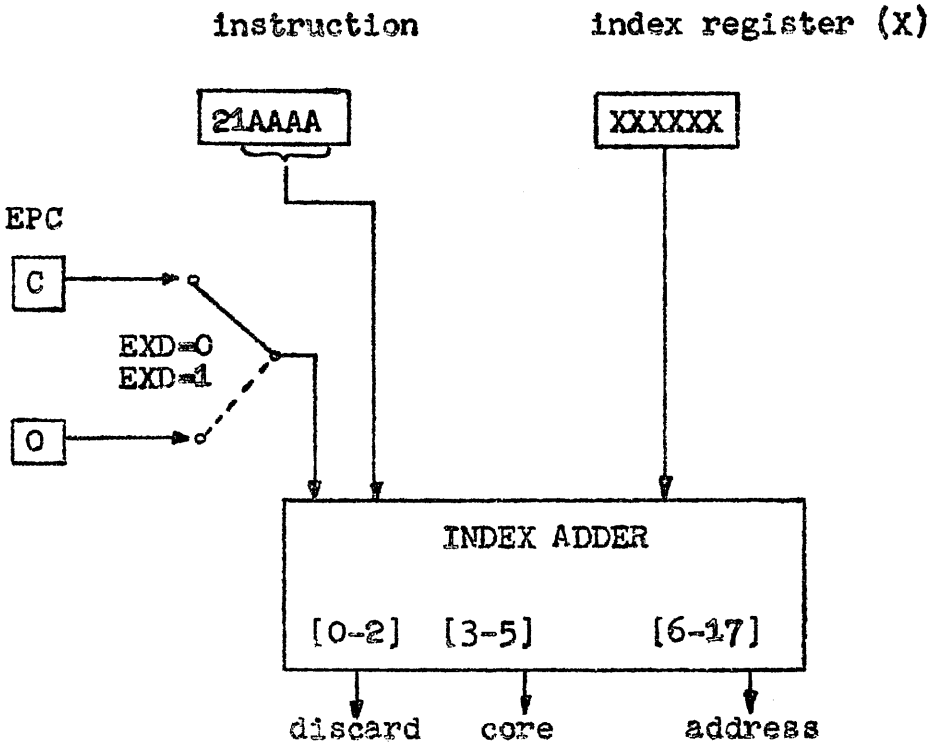
2.2 Indexed Addresses

In various cases addresses will be modified by addition of the contents of the index register (X). Depending on the address mode, indexing can apply to either the address part of an instruction or the last indirect address in a defer chain. The index adder is 18 bits long, having the 18-bit index register as one input and a 15-bit address as the other. The sum is computed in the current arithmetic mode (one's or two's complement). In one's complement mode, minus zero is changed to plus zero. The sum is then truncated to 15-bits for use as the data address.

Indexing alone does not lengthen the execution time of an instruction.

2.2.1 Indexing an Instruction

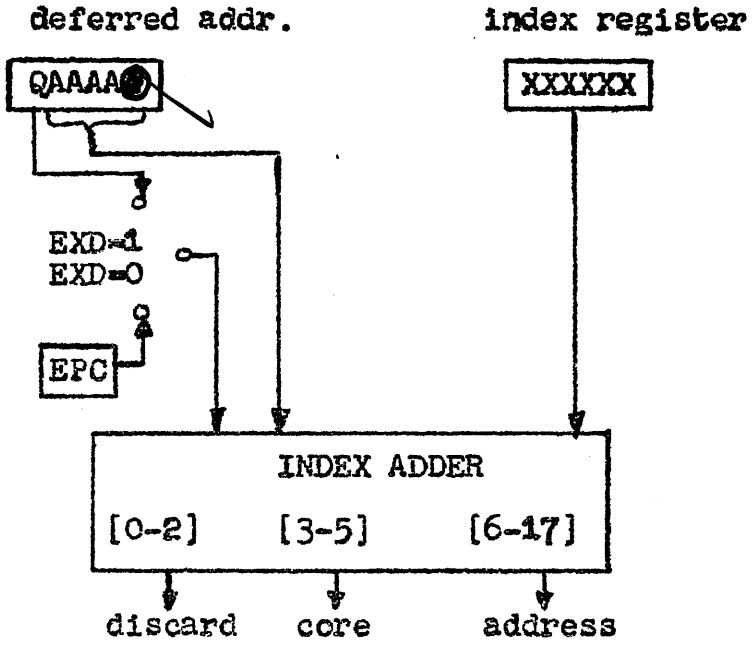
If the machine is not in extend mode, index sums are relative to the current core. That is, the EPC is added in to bit positions 3 to 5. If EXD is on, zero is added into these positions so that the index sums are absolute (relative to core 0).



2.2.2 Indexing a Deferred Address

A deferred address may also be indexed. If the computer is not in extend mode, the address part of the deferred address and the EPC are used. If EXD is on, the low 15 bits of the deferred address are used.

Indexing of indirect addresses applies only on the last defer cycle.



2.3 Address Modes

The PDP-1 has four address modes -- normal, base, index (first), and defer (first). The AMD and AEF flip-flops determine which mode the machine is in.

Regardless of the address mode, jump-type instructions (jmp, jsp, jda, jdp) behave as if they are in normal mode (cf.). jda (Op. Code 17) always behaves as if its i-bit were off.

The address mode may be temporarily changed by the aam instruction, 770056, which turns on the address alternation flip-flop AAL. AAL is automatically cleared at the end of the following instruction. The effect of an aam depends on the address mode and will be discussed separately for each of the four.

Note a jdp, jda, jmp, or jsp immediately following an aam will be executed in base mode, since the nominal mode for these instructions is normal mode.

The cal instruction always behaves the same (goes to core 0, location 100) regardless of the address mode.

xct instructions behave like any other addressable instruction with respect to address modes and being preceded by aam instructions. The instruction xct'd is always in the nominal mode, as determined by AMD and AEF. If there were an aam over the xct, it would affect only the xct, and not the instruction xct'd.

Two aam's in a row leave AAL off.

2.3.1 Normal Mode

This is the only mode a normal PDP-1 operates in. The machine will be in this mode if any of the console buttons except STOP or CONTINUE is pressed. The timesharing system initializes processes to be in normal mode.

Normal mode is entered by executing a nam instruction, which clears both AMD and AEF. In normal mode addresses are direct if the i-bit is off and indirect if it is on. See section 2.4.1 for examples.

An aam in normal mode causes indexing to be applied regardless of whether the i-bit is on or not. Indexing happens before indirecting if both are specified. See section 2.4.2 for examples of alternated normal mode.

2.3.2 Base Mode

In base mode the index register usually points to the base of a "private block" associated with the specific process that is executing reentrant code. All addressable instructions (except jump-type instructions) are indexed into the private block.

Base mode is identical to alternated normal mode. In base mode AMD is off and AEF is on. Base mode is entered by executing a bam instruction. See section 2.4.2 for examples.

Alternated base mode is identical to normal mode. That is, never index and indirect only if the i-bit is on. See 2.4.1 for examples.

2.3.3 Index and Defer Modes

An iam instruction puts the PDP-1 in index mode (AMD=1, AEF=0), and a dam instruction puts it in defer mode (AMD=1,AEF=1). The only difference in these two modes is the order in which indexing and indirecting are done if both are specified. Index mode indexes first, while defer mode defers first.

In unalternated index or defer mode addresses are direct if the i-bit is off and indexed if it is on. See section 2.4.3 for examples.

When in alternated index or defer mode, instructions always defer and if the i-bit is on, they also index. The order is determined by the mode. Section 2.4.4 contains the examples of alternated index mode, and section 2.4.5 has the alternated defer mode examples.

2.3.4 Summary of Address Modes

Mode	AMD	AEF	AAL	Indirect	Index
normal	0	0	0	if i-bit on	no
"	0	0	1	if i-bit on	yes (index first if both)
base	0	1	0	if i-bit on	yes (index first if both)
"	0	1	1	if i-bit on	no
index	1	0	0	no	if i-bit on
"	1	0	1	yes	if i-bit on (index first)
defer	1	1	0	no	if i-bit on
"	1	1	1	yes	if i-bit on (defer first)

2.4 Examples

2.4.1 Normal Mode or Alternated Base Mode

```
        nam                                     bam
        lac 30          OR                      aam
                                                lac 30
          .                                       .
          .                                       .
          .                                       .
30/     105                             30/     105
```

The accumulator is loaded from location 30 of the current core. No index or defer happens.

```
        nam                                     bam
        lac i 30        OR                      aam
                                                lac i 30
          .                                       .
          .                                       .
          .                                       .
30/     10105                           30/     10105
105/     1234           105/             1234
1234/       3           1234/               3
10105/      6           10105/              6
```

If the computer is not in extend mode, A is loaded with 3 after two defers. If EXD is on, a single extended defer happens, picking up the 6 from core 1.

2.4.2 Base Mode or Alternated Normal Mode

Assume X has +10 in it.

```
      bam                                    nam
      lac 30          OR                     aam
                                             lac 30
        .                                      .
        .                                      .
        .                                      .
30/   105                              30/   105
40/   21                               40/   21
```

The effective address is 30+(contents of X) = 40. Thus, 21 is put into A.

Assume X contains 777770.

```
      bam                                    nam
      lac 70          OR                     aam
                                             lac 70
        .                                      .
        .                                      .
        .                                      .
60/   105                              60/   105
61/   144                              61/   144
70/   -10                              70/   -10
```

The effective address is 70+(contents of X). This sum differs in one's and two's mode. In one's mode, the effective address is 61 and therefore 144 is put into A. In two's mode, the effective address is 60 and accordingly 105 is placed into A.

Assume X has 10100 in it and the same program is in core 2. If the machine is not in extend mode, the index address is relative to the current core (2). Thus, the effective address is 20000+10100+30=30130 or location 30 of core 3. If EXD is on, the effective address is 10130 or location 130 of core 1.

Consider the following program in core 2 with 10100 in X.

```
10130/   31234                    10130/   31234

                                  20007/   nam
20010/   bam                     20010/   aam
         lac i 30                          lac i 30
          .                                 .
          .                                 .
          .                                 .
30130/   11234                    30130/   11234
31234/   62345                    31234/   62345
32345/   6                       32345/   6
```

Out of extend mode, a defer chain is started at (address part) + (index register) + (current core) = 130 of core 3. Two defers happen and 6 is finally loaded into the accumulator.

In extend mode a single extended defer is done through core 1 location 130. This location points to core 3 location 1234, so that 62345 is put into A.

2.4.3 Unalternated Index or Defer Mode

```
        iam         /or dam
        lac 30
         .
         .
         .
30/     105
```

No index or indirect modifications happen. The accumulator is loaded with 105.

Assume the following program is in core 2 and X has 10100 in it.

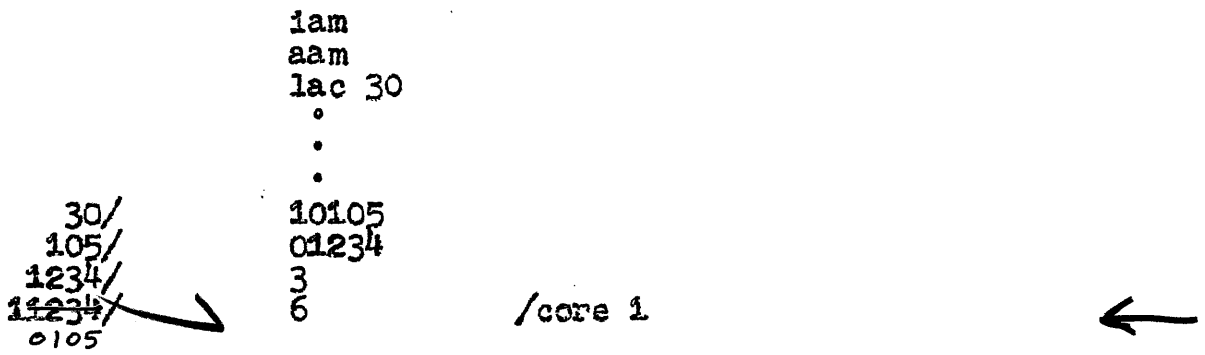
```
10130/    6

20010/    iam         /or dam
20011/    lac i 30
           .
           .
           .
30130/    11
```

Only indexing can happen. Out of extend mode the effective address is (contents of X) + (address part) + (current core) = 30130 or location 130 of core 3. Eleven is put into A.

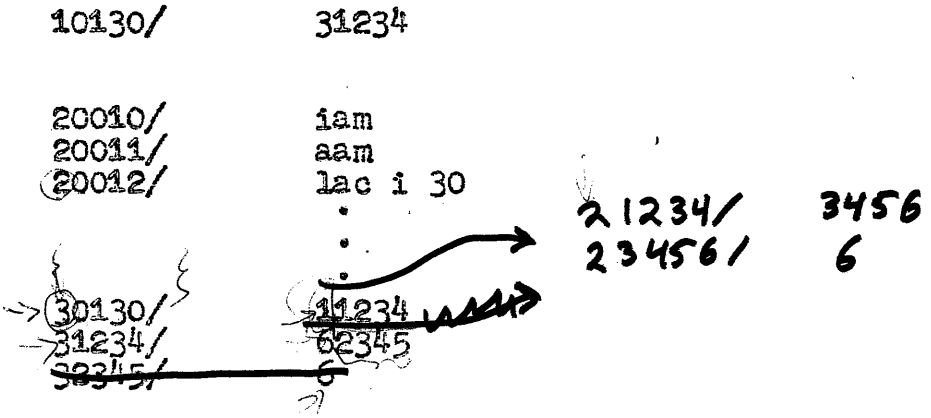
In extend mode the reference will be to 10130 or location 130 of core 1. Six will be put into A.

2.4.4 Alternated Index Mode



No indexing occurs, but the instruction defers through location 30 of the same core. If EXD is on, this is an extended defer and the accumulator gets loaded with 6. If EXD is off, two defers happen and A picks up the 3 from location 1234.

Assume X has 10100 in it and the following program is in core 2.



In index mode indexing happens before defering. Thus, out of extend mode a two-level defer chain is started at location 130 of core 3. This ends by loading the accumulator with 6. In extend mode a single extended defer is done through 130 of core 1, and A is loaded with 62345.

2.4.5 Alternated Defer Mode

```
                dam
                aam
                lac 30
                  .
                  .
                  .
    30/         10105
   105/         01234
  1234/         3
 10105/         6              /core 1
```

No indexing occurs. Out of extend mode, the instruction defers through location 30 and again through 105, resulting in 3 being put into the accumulator. In extend mode, a single defer causes the 6 in location 105 of core 1 to be loaded.

In the next program assume that core 2 is current and the index register contains 10100.

```
 11234/     17730

 20010/     dam
 20011/     aam
 20012/     lac i 30
              .
              .
              .
 20030/     11234
 21234/     2345
 21334/     3

 32445/     6
```

In alternated defer mode defers happen before indexing. Thus, out of extend mode, the lac i 30 will defer twice, resulting in 2345 which will be added to the index register and current core(2), resulting in a final effective address of core 3 location 2445. Six would be loaded into A.

With EXD on, a single defer happens through 30 of core 2. Then 11234 is added to the index register, giving a final address of 21334. Therefore, the accumulator is loaded with 3.

2.5 Instructions that Affect Effective Address Calculation

Mnemonic	Op. Code	Function
nam	770052	enter normal address mode
bam	770053	enter base address mode
iam	770054	enter index (first) address mode
dam	770055	enter defer (first) address mode
aam	770056	cause 1-instruction mode change
eem	770046	enter extend mode
lem	770047	leave extend mode
lpf	770051	loads AMD, AEF, AAL from I[0-2]. See section 1.3.4 However- AAL will be turned off at the end of this instruction.

2.6 Sample Programs

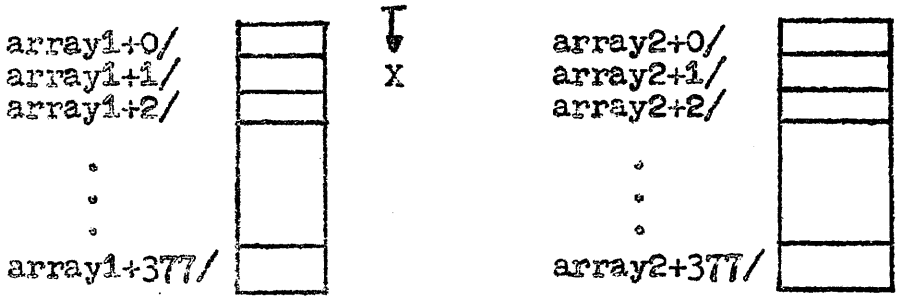
The following are examples of uses of the various address modes and the index register.

2.6.1 Exchange Array Contents

The following program segment exchanges the contents of two arrays, named array1 and array2. Both are 400 words long. In this example the index register is used both as a counter and an increment into the arrays.

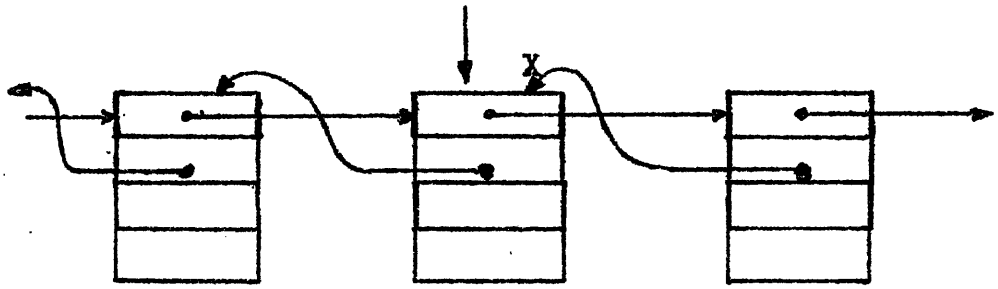
```
/runs in either iam or dam
/the -400 below should be considered in the same arithmetic
/  mode as that in which the program runs

exchng,     lxr (-400           /minus the length to X
next,       lac i array1+400    /pick up the words
            lio i array2+400
            dio i array1+400    /put back exchanged
            dac i array2+400
            SXXZ                /add 1 to X,skip if up to 0
            jmp next            /not done
done,
              .
              .
              .
```



2.6.2 Unchain a Data Block

In a certain application data blocks are chained as shown below. Word zero is a forward pointer. Word one is a back pointer. The subroutine given below will delete the block pointed to by the index register. In this situation the index register always contains the address of the base (word 0) of a block.



block to be deleted

```
/jdp rmb in iam
/X points to block to be deleted

rmb,    0
        lio i 0     /I points to block on right
        aam
        dio i 1     /index then indirect to fix left block
        lac i 1     /A points to left block
        TIX         /move to right block
        dac i 1     /fix back pointer
        jmp i rmb   /return
```