PDP-11/40 Technical Memorandum # 18

TITLE: The Execute Instruction

AUTHOR(S): Ad van de Goor

INDEX KEYS: Execute
Interruptability
Addressing Modes

DISTRIBUTION LIST: PDP-11/40 Group
PDP-11/60 Architecture Review Committee
Jud Leonard
Jeff Scott
Richard De Morgan

REVISION: None

OBSOLETE: None

DATE: September 21, 1970

0.0 ABSTRACT

(This is the abstract of PDP-11/40 Technical Memorandum #18, dated September 21, 1970, total pages 7 .)

The use of the execute "XCT" instruction is discussed and from that an implementation derived.

Interruptability considerations resulted in a non-interruptable execute sequence. (This was the simplest solution hardware wise.)

Considerations of the addressing modes used by the XCT instructions led to the rule that when the -(R), (R)+ and (R)A modes are used, the instruction to be executed is "thought of" as being one 16-bit word long, independent of the actual length of the instruction to be executed.

In order to have no surprises or side effects in the instructions to be executed, the machine is required to have two program counters:

1) The PC, which is the regular program counter. Upon completion of the XCT sequence, it usually points to the instruction following the XCT instruction.

2) The dummy program counter "DPC", which is only used for the address computation in the instruction to be executed.

The only instructions which cannot be executed are Execute and Repeat.

1.0 BASIC CONSIDERATIONS

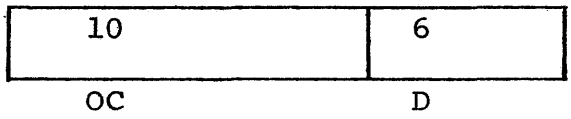
In order to determine the way the Execute "XCT" instruction should be implemented, the usage of the XCT instruction should be given some attention. The main reasons for its existence are twofold:

1. To facilitate the writing of reentrant code. This is accomplished by having the "impure" part of the code stored as data. Instruction stored that way can be executed through the XCT instruction. The net effect of using XCT this way should be the same as if the executed instruction would have been executed in line, i.e. in the place of the XCT instruction.

2. To allow for the execution of a selected instruction. The effect of using XCT this way is that of the execution of a one word subroutine (except for the RTS instruction).

2.0 A POSSIBLE IMPLEMENTATION

The format of the Execute "XCT" instruction is as shown below:



The OC field specifies the XCT instructions. The D field is a regular destination field pointing to the instruction to be executed. Interaction with the to-be-executed instruction is possible when the (R)+, -(R), and @(R)+ modes are used in the D field of the XCT instruction.

The evaluation of the D field is done in the same way as a JMP instruction, i.e. the instruction to be executed starts at the location indicated by the effective address of the D field of the XCT instruction.

2.1 INTERRUPTABILITY

There are three alternative ways in which an instruction sequence can be interrupted. (The following discussion is only with relevance to the XCT instruction.)

1) Interrupt in the middle of the instruction and resume where left off.

   The XCT sequence consists of at least two instructions (namely the XCT instruction itself and the instruction to be executed). An interrupt in the middle of this sequence which allows for continuation from that point requires two addresses to be stored:

   a) The address of the next instruction, i.e. the instruction following the completion of the XCT sequence.

   b) The address of the instruction which had to be executed next in the XCT sequence.

   Besides having two addresses to be saved upon an interrupt, a status bit has to be set indicating that this happened. Considering the complexity of having to deal with a possible third word (i.e. the PC, PS, and the second address) and a status bit indicating the presence of the third word this alternative has to be rejected.

2) Abort instruction and start over. Because of the address modifications which can occur when the -(R), (R)+ and @(R)+ modes are used on the 11/40, starting over is not possible.

3) Finish Current Instruction. This means that interrupts are only allowed between complete instructions or complete instruction sequences.

   Because the other two alternatives are too complicated or not possible, this will be the way XCT instructions will be interruptable (namely they are not interruptable).

Having chosen the third alternative, chains of XCT instructions (like allowed in the PDP-10) should not be allowed in order to cut down the "interrupt response" time. Considering this constraint, an XCT sequence consists of two instructions only. Furthermore, it should be remembered that the XCT instruction can add a three memory cycle time (when the XCT instruction uses the @(R)+ or the @A(R) addressing modes) to the execution time of the instruction to be executed, as far as the "interrupt response" is concerned.

3.0 ADDRESSING MODES OF THE XCT INSTRUCTION

There are eight possible addressing modes because of the D field of the XCT instruction.

3.1 "R" - Register Mode

In this mode the first word of the instruction to be executed is located in the 16-bit register denoted by R. In case the instruction to be executed consists of multiple words, the second, third, etc. words are located in the registers R+1, R+2, etc.

It is interesting to note that in this mode "register with register" indexing is possible when the executed instruction uses the mode A(R), (R)A, or @A(R).

3.2

"@(R)" - Indirect Register Mode
"A(R)" - Indexed Mode
"@(R)+" - Auto Increment Deferred Mode
"@A(R)" - Indexed Deferred Mode

The above four modes are handled in the usual manner. In the case of the A(R), @A(R) and @(R7)+ modes, the PC is incremented with 2.

3.3

"(R)+" - Auto Increment Mode
"-(R)" - Auto Decrement Mode
"(R)A" - Adjusting Index Mode

These three modes have all one thing in common, namely that their operation depends on the data type they operate upon. (The to-be-executed instruction has to be considered as data as far as the XCT instruction is concerned.) When the above modes are normally used, the length of the data the instruction operates upon is implied in the instruction. The XCT instruction, however, does not contain any informa-

tion concerning the length of its data (i.e. the instruction to be executed).

The adjustment by a variable amount could be desirable if, for example, the (R)+ mode were used and a sequence of instructions had to be executed through the XCT instruction. R would then automatically point to the next instruction.

The idea of an adjustment by a variable amount has to be rejected because of the following reasons:

1) In case of the -(R) mode, the adjustment is impossible because the length of the to-be-executed instruction is unknown and cannot be determined.

2) In case of the (R)+ mode, the following sequence of instructions could occur.

```
XCT  (Rx)+ ⟹ MOV  (Rx)+,A(Ry)
XCT  (Rx)+ ⟹ MOV  #5, Rx
```

In order to have the MOV instructions work correctly, Rx in the XCT instruction has to be adjusted before the MOV instruction is executed. This means that the to-be-executed instruction (MOV in this case) has to be read from memory, its length has to be analyzed, Rx has to be adjusted, and finally the MOV can be executed.

Because of the problems above, it is suggested to have a fixed, rather than a variable adjustment. This adjustment should be such that the XCT instruction thinks it operates on data of one 16-bit word (i.e. when the -(R) and (R)+ modes are used the adjustment is 2, when the (R)A mode is used L=2).

4.0 INSTRUCTIONS TO BE EXECUTED

Because of the requirements of section 1.0, the PC is handled in such a way that upon completion of the XCT instruction sequence, the PC points to the instruction directly following the XCT instruction, unless a transfer of control instruction has been executed (e.g. a Branch, Jump, etc.).

In trying to determine how the PC should be handled, a few coding examples will be analyzed.

1) XCT Dest MOV #5, A(R)

2) XCT Dest ADD #8, PC

3) XCT Dest JSR R, Dest

In example 1 we want the next instruction to be executed to be the one following the XCT instruction. In example 2 we want 8 to be added to the "real" PC, i.e. the next instruction to be executed will be 4 words down from the XCT instruction. The implication of this is that the machine needs two program counters.

1) The program counter "PC". This is the regular Program counter. It is used exclusively in the XCT instruction and in the executed instruction only in the execution phase, i.e. not in the address computation phase.

2) The dummy program counter "DPC". This DPC is only used in the address computation steps of the to-be-executed instruction. Its initial value is the address evaluated in the XCT instruction.

Having these two program counters, the effect of executing a JSR,as in Example 3, will be quite clear. The program counter (i.e. return address) saved is the PC (rather than the DPC). This way the current subroutine calling conventions are obeyed.

4.1 INSTRUCTIONS WHICH CANNOT BE EXECUTED

There is a group of instructions which are not allowed to be executed.

1) Execute "XCT". This is because of the interrupt response time as discussed before.

2) Repeat "RXX". This should be disallowed because of the to-be-expected hardware complexities.

4.2 QUESTIONABLE INSTRUCTIONS WHICH CAN BE EXECUTED

There is a group of instructions which affect the flow of control in a program. All of these (see list below) can be executed (i.e. through the XCT instruction).

1) Branch
2) Jump
3) Subroutine Call "JSR"
4) Substract and Branch "SOB"
5) Emulator Trap "EMT"
6) Trap
7) Return from Interrupt "RTI"
8) Return from Subroutine "RTS"