PDP-11/40 Technical Memorandum # 17

TITLE: The Repeat Instruction

AUTHOR(S): Ad van de Goor

INDEX KEYS:
Repeat Count
Repeat Conditions
Resumption
Reentrance
Interruptability

DISTRIBUTION:
PDP-11/40 Group
Bruce Delagi
Jud Leonard
Richard DeMorgan
Jeff Scott
Jack Burness

REVISION: None

OBSOLETE: None

DATE: September 2, 1970

0.0 Abstract

The desirability of a conditional repeat instruction has been demonstrated on several occasions. The repeat instruction provides for the shortest possible program loop (namely a single instruction), besides that it seems to have a big market appeal.

Because this instruction is considered for the 11/25, the possible 11/40 implementations are considered for compatibility reasons.

The problems with this instruction are two-fold:

1) How to implement it, considering implementation cost and required OP code space.

2) Which instructions cannot be repeated,or repeated with restrictions.

1.0 Implementation Aspects

The Univac 1108 has a repeat operation implemented in the following way:

1) A special repeat register is used to contain the repeat count.

2) Special, one word, instructions make use of the repeat operation (e.g. Search for Equal, Search for Greater, etc.). The repeat action is implied in the instruction.

3) When the repeat condition (e.g. Equal, Greater, etc.) is satisfied, the next instruction is skipped. Otherwise, the next instruction is executed.

Considering the limited PDP-11 OP code space, special instructions, making use of the repeat operation, are difficult, if not impossible, to implement. An alternative way is discussed below. It essentially makes every repeat operation a two instruction sequence.

The programmer will experience this 2 instruction sequence in the following way:

```
RXX   cnt
Instruction to be repeated
```

The first instruction "RXX cnt" (which is a special instruction) is the repeat instruction where "R" is the mnemonic for repeat, "XX" stands for the repeat condition and "cnt" stands for the repeat count.

The second instruction (which can be a regular PDP-11 instruction) is the instruction to be repeated. (Note: only a single instruction will be repeated because of cost considerations.) The repeat operation stops when the "cnt" is zero or the condition "XX" is satisfied or both (i.e. stop if cnt=0 or XX =true). The cnt is considered a positive integer and is decremented for every execution of the "instruction to be repeated."

The problems concerning the implementation are the following:

1) What are the conditions "XX."

2) How is the repeat count specified.

3) How can the stop reasons be determined.

1.1 The repeat conditions "XX."

Rather than introduce a set of new conditions, it is suggested to use the branch conditions for this as well as their mnemonics. They are known and considered adequate. The 11/40, until now, has 16 branch conditions, 15 of which are the same as those for the 11/20. The 16th branch condition is "BSF" (branch if top of the stack is false) followed by a pop of the stack (see PDP-11/40 Technical Memo #15). This branch condition is not expected to be very useful and hard to implement. It is therefore suggested to make this an illegal branch condition.

1.2 Specification of the Repeat Count "CNT"

Three requirements should be made concerning the handling of the "cnt."

1) Upon a stop, because the condition XX=true, it should be possible to resume the repeat operation as if the stop had not occured.

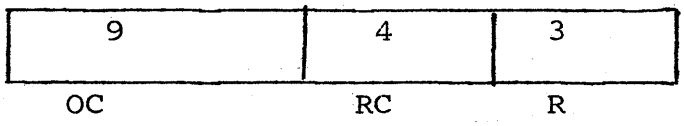
2) Stops should be implemented in such a way that reentrant coding is not excluded.

3) The "setup" of the repeat instruction should be as small as possible. This applies especially to the count "cnt."

Until now, two basic solutions have been found and are discussed below.

1.2.1 A Register Specifies CNT

The format of the repeat instruction for this case will be as follows.



The repeat count "cnt" is specified by the general purpose register indicated by the 3-bit "R" field. The repeat condition XX is specified by the 4-bit "RC" field.

When the repeat operation stops or is interrupted, the remainder of the cnt is stored in the register specified by the 3-bit "R" field. This way requirements 1 and 2 of section 1.2 are satisfied.

With respect to the third requirement, the following sequence of instructions can be considered quite common.

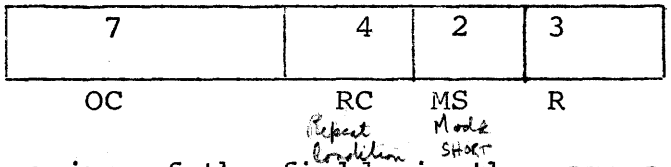
```
MOVE     R,   -(SP)     /save register
MOVE    #cnt, R         /load cnt in R
Repeat
Instruction to be repeated
MOVE     (SP)+,  R      /restore register
```

This shows that 3 instructions, occupying 4 words are required for "setup."

It should be noted that the register R, containing the cnt, cannot be used by the "instruction to be repeated" unless the repeat count is updated in R.

1.2.2 Effective Address "EA" Specifies cnt

The format of the repeat instruction for this case is shown below:



The meaning of the fields is the same as those shown in section 1.2.1 except for the 2-bit short mode field "MS."

The 11/40 addressing modes differ from the 11/20 addressing modes. The first four 11/40 modes are not only used to specify addresses for data, but also to specify the shift/rotate count in multiple shift/rotate instructions.

These four modes are (see memo #15):

NUMBER	MODE	EA	CNT RANGE
0	R	R	0 to 7
1	@ R	(R)	-2↑15 to 2↑15 -1
2	A(R)	A+(R)	-2↑15 to 2↑15 -1
3	@(R)+	((R))	-2↑15 to 2↑15 -1

Mode #0 allows for small values of cnt to be specified in

the instruction word itself (i.e. in 16 bits). Mode #1 allows the count to be specified in a register, like in section 1.2.1. Mode #4 allows the count to be specified by the top word of a stack or the word trailing the repeat instruction (when @(R7)+ is used).

Looking at the first four 11/40 addressing modes, it can be concluded that only mode 2 "A(R)" differs from the 11/20-11/25 mode 2 "(R)+". Upward compatibility can be guaranteed by interpreting mode 2 on the 11/25 differently (i.e. as A(R) rather than (R)+) for the repeat instruction or declaring this mode illegal.

When the repeat operation stops, because XX=true, the remainder of the count can be saved or ignored. The table below shows some alternatives.

	ACTION WHEN XX=TRUE			
Mode	Alt. 1	Alt. 2	Alt. 3	Alt. 4
R	cnt→-(SP)*	cnt→-(SP)*		
@R	cnt→-(SP)*	cnt→R	cnt→R	cnt→R
A(R)	cnt→-(SP)*	cnt→-(SP)*		
@(R)+	cnt→-(SP)*	cnt→-(SP)*	cnt→-(SP)**	cnt→-(R)***

NOTE:
* only when cnt ≠ 0
** only when cnt ≠ 0 and R=R6
*** only when cnt ≠ 0 and R≠R7.

Alternative 1 satisfies requirement 1 (i.e. resume after stop) only when the @(R)+ mode is used with R=R6.

Alternative 2 satisfies requirement 1 when the @R mode is used and when the @(R)+ mode is used with R=R6.

Alternative 3 satisfies requirement 1 when the @R mode is used and when the @(R)+ mode is used with R=R6, i.e. under the same conditions as alternative 2. The difference between alternative 3 and alternative 2 is that when requirement 1 in alternative 3 is not satisfied (i.e. for the modes R, A(R) and @(R)+ with R≠R6) the remainder of the repeat count is not saved.

Alternative 4 is like alternative 3 except that requirement 1 is satisfied for the @(R)+ mode under more general conditions, i.e. for R=R1, R2, R3, R4, R5, and R6 rather than R=R6 only.

The repeat operation can be interrupted by storing the remainder of the cnt as the third word on the stack (i.e. after the PC and PS words). When the mode "@R" is used, the remainder of the cnt could be stored in R upon interrupt.

The amount of OP code space required for this solution is four times as much as that of the solution of section 1.2.1. It is still only 1/8 of an OP code space, however.

If the 11/25 would implement the method of section 1.2.1, compatibility with this method can then be guaranteed for alternatives 2, 3, and 4.

1.3 Stop Reasons

A repeat operation can stop for the following three reasons:

1) XX= True and cnt $\neq$ 0

2) XX= True and cnt = 0

3) XX= False and cnt = 0

For the programmer it can be very important to know for which of the above three reasons the program stopped.

On the Univac 1108 this problem was solved by having the next instruction skipped when the repeat operation stopped for reasons 1 and 2 as listed above. Unconditional repeat operations, like block transfer (e.g. a repeated move) never skipped.

1.3.1 Skip Method

This method is similar to the one used for the Univac 1108. A disadvantage of this method is that it introduces a new concept in the PDP-11 architecture, namely skipping. This concept, however, may also be used by some other instructions to be added (like the P and V interlock instructions).

For the 11 family it is suggested to have the repeat operation skip one word when a stop occurs because of reasons 1 and 2, and no skip when a stop occurs because of reason 3. For unconditional instructions a skip would mean the waste of a word; it is therefore suggested never to skip when the condition "Always" is used (i.e. the condition associated with the branch instruction BR). A single skip of one word is advisable because: a) A variable length skip would take more OP code space (to specify the length), and b) A fixed length skip of more than one word would lead to a waste of core memory in many cases.

Because a single skip cannot distinguish between reasons 1 and 2, the following requirements have to be made of the suggested solutions of section 1.2.

For solution 1.2.1, when the repeat operation starts with cnt =0, the "instruction to be repeated" should not be executed and no skip should take place (i.e. the stop reason should be that of number 3).

For solution 1.2.2 when the repeat operation starts with cnt =0, the "instruction to be repeated" should not be executed, no skip should take place, and if the zero cnt was specified through the @(R)+ mode, the auto increment should take place. For those alternatives and mode combinations where a resume after stop is possible, the footnotes "*", "**", and "***" should be changed such that when the repeat operation stops because of reason 2 (XX=true and cnt =0). The remainder of the count (i.e. cnt =0) should be stored. This means that the footnotes should read as follows:

* Only when cnt $\neq$ 0 or when cnt =0 and XX = True (i.e. stop reasons 1 and 2)

** Only when R=R6 and (cnt $\neq$ 0 or cnt =0 and XX = True)

*** Only when R$\neq$R7 and (cnt $\neq$ 0 or cnt = 0 and XX = True)

Using the skip method, the condition code of the machine can be handled in the "normal" way, i.e. according to the "instrution to be repeated." When the initial cnt = 0 the"instruction to be repeated" is not executed and therefore the condition code will not be affected.

1.3.2 The Condition Code Method

Here the condition code of the machine is used to indicate the stop reason which then can be tested with the regular branch instructions. In order to be able to branch on all stop conditions, their inverse conditions and some combinations, it is suggested to set the Z, N, and C condition code bits, upon a stop, according to the table below. The V bit is not affected. The exact way of handling the condition code is not very important as long as the major conditions can be tested. An example is given below.

STOP REASON	COND. CODE			BRANCH ON COND.		BRANCH ON INV. COND.	
	C	Z	N				
1. XX=true & cnt ≠0	1	0	0	Z=0	BNE	Z=1	BEQ
2. XX=true & cnt =1	1	1	1	N=1	BMI	N=0	BPL
3. XX=false & cnt =0	0	1	0	C=0	BCC	C=1	BCS
1 or 2				C=1	BCS	C=0	BCC
2 or 3				Z=1	BEQ	Z=0	BNE

A disadvantage of this method is that the condition code has to be set in a special non-standard way, which means some more hardware and, perhaps more important, that the condition code has a different meaning after a repeat instruction then after all other instructions. This means that the above table has to be memorized (or at least consulted) when repeat is used.

2.0 Instructions to be Repeated

The instructions to be repeated can be executed in two different ways.

1) Fetched from memory for every execution

2) Fetched from memory only once for the whole repeat operation, unless interrupted. Upon interrupt the instruction to be repeated has to be refetched.

The advantage of the second scheme is a much higher execution speed, the disadvantage is that self modifying instructions will not work. The latter is not a severe limitation, and does not cancel the advantage of the higher speed.

It is suggested, for speed reasons, that operands referenced by the repeated instruction are refetched and rewritten only if address changes occur.

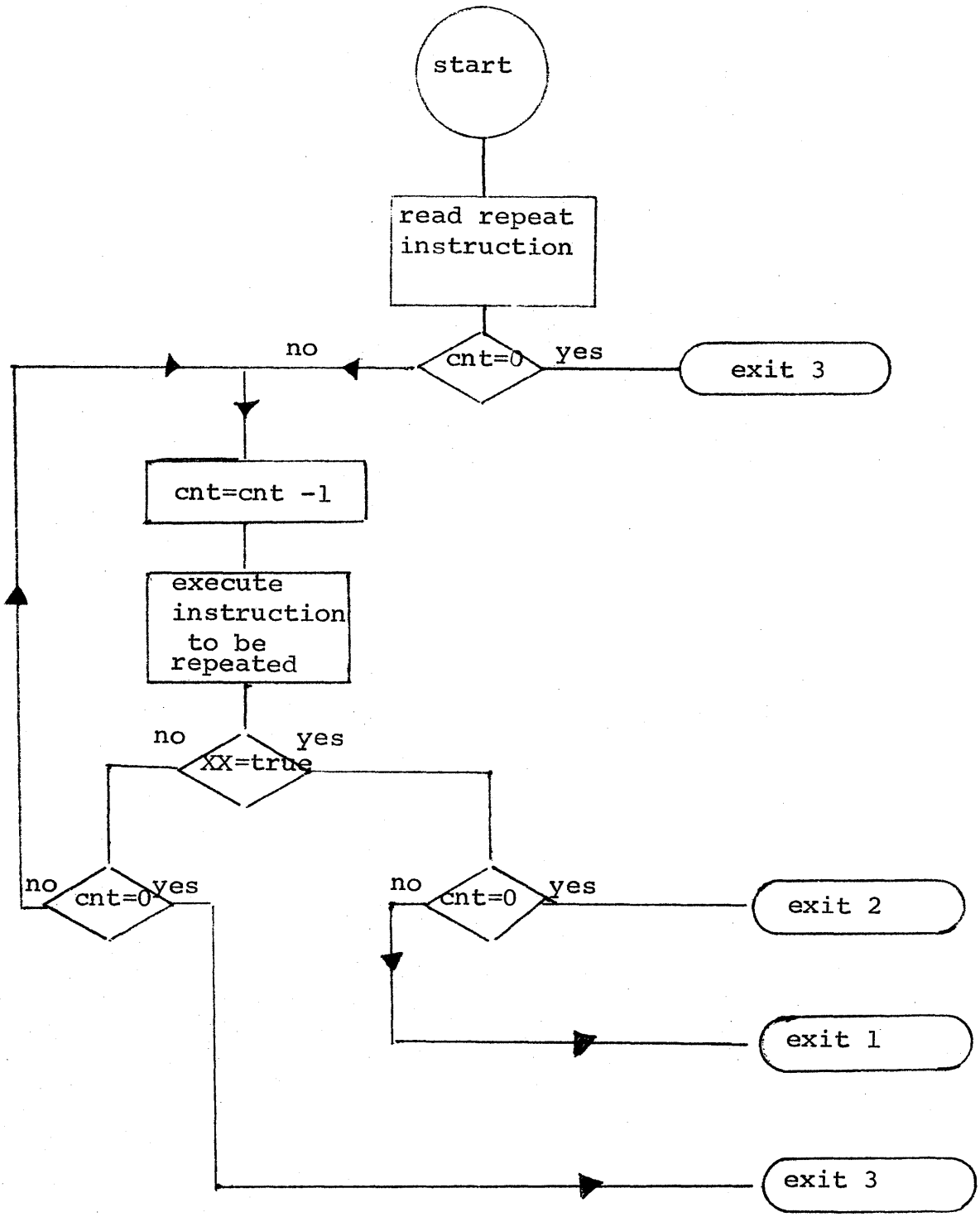
Upon interrupts, the saved PC will point to the repeat instruction, rather then to the instruction to be repeated in order to allow for easy resumption.

Some questions which have to be solved are:

1. Are multiple word length instructions (i.e. 32, 48 and 64-bit) allowed to be repeated because of problems in handling the PC?

2. Where does the PC point to after the repeat operation stops? If it is forced to point to the instruction directly following the instruction to be repeated or the word thereafter (assuming the skip method) certain groups of instructions are made no-ops automatically (e.g., Branches, Jumps, SOB instructions, etc.). If JSR instructions have to be repeated, the parameter passing will be more difficult then because the first instruction following the JSR has to be a Branch around the parameters following the call (assuming there are parameters) which is non-standard.

A flow chart of the conditional repeat instruction could be as follows. (The count is shown in "cnt", the condition in "XX", and the exit numbers are those of the 3 stop conditions of section 1.3.)


start
read repeat instruction
cnt=0
yes
exit 3
no
cnt=cnt -1
execute instruction to be repeated
XX=true
yes
no
cnt=0
yes
no
cnt=0
yes
exit 2
exit 1
exit 3

3.0 Conclusions

Looking at some PDP-11/20 programs, it can be noted that a repeat with a small count (typically less than 8) would be very useful. This suggests that the count should be determined according to method 1.2.2, i.e. through an effective address. Whether any of the four suggested alternatives should be used or still another alternative for storing the remainder of the count when the repeat operation stops, is still a point of discussion.

As far as the stop reasons of section 1.3 are concerned, it should be noted that the skip method will be easier to understand by the programmer. The condition code method has the advantage that upon stop reason #2 (i.e. XX=True and cnt =0) the remainder of the cnt (which is =0) does not have to be saved in order to satisfy the resumption requirement (i.e. requirement #1). Another advantage is that any of the stop reasons and any of their combinations can be tested for.

Before a final decision can be made, some more (your!) inputs are very desirable.

djc