PDP-11/40 Technical Memorandum # 14

Title: Register Layout for 11/40 and Up

Author(s): Ad van de Goor

Index Keys: Register Layout
Overlapping
Register-Code Overlay

Distribution: PDP-11/40 Group
PDP-11 Coordinating Committee
Bruce Delagi

Revision: None

Obsolete: None

Date: July 14, 1970

0.0 Abstract

Because the 11/40 might, and bigger versions of the PDP-11 family will, have 32-bit integer and floating-point arithmetic capabilities, the need for more general purpose hardware registers is quite clear.

other – The competition of the bigger PDP-11's (e.g., PDP-11/60) has typically sixteen 32-bit general purpose registers.

Four methods of adding, in a compatible way, new registers to the PDP-11/20 architecture are discussed. The last method, section 1.4 "The Disjoint Scheme" is preferred because it has most of the advantages and satisfies implementation constraints.

The issue of multiple register blocks will be discussed in another memo.

1.0 Register Layout Schemes

Until now the author has come up with four schemes for adding, in a compatible way, more addressable hardware registers to the PDP-11/20 architecture. In case a better scheme is overlooked or any other shortcoming of this memo, the reader is urged to make his comments, suggestions, etc. known.

The register layout schemes discussed are:

1. The variable length scheme
2. The completely overlapping scheme
3. The partially overlapping scheme
4. The disjoint scheme

1.1 The Variable Length Scheme

With this scheme the machine has a total of 8 general purpose registers, see Figure 1-1. Register R6 and R7 are the 16-bit dedicated registers SP and PC respectively. The registers R0-R5 are each as long as the longest data type implemented on the machine. This could be floating double and could be 64 bits.

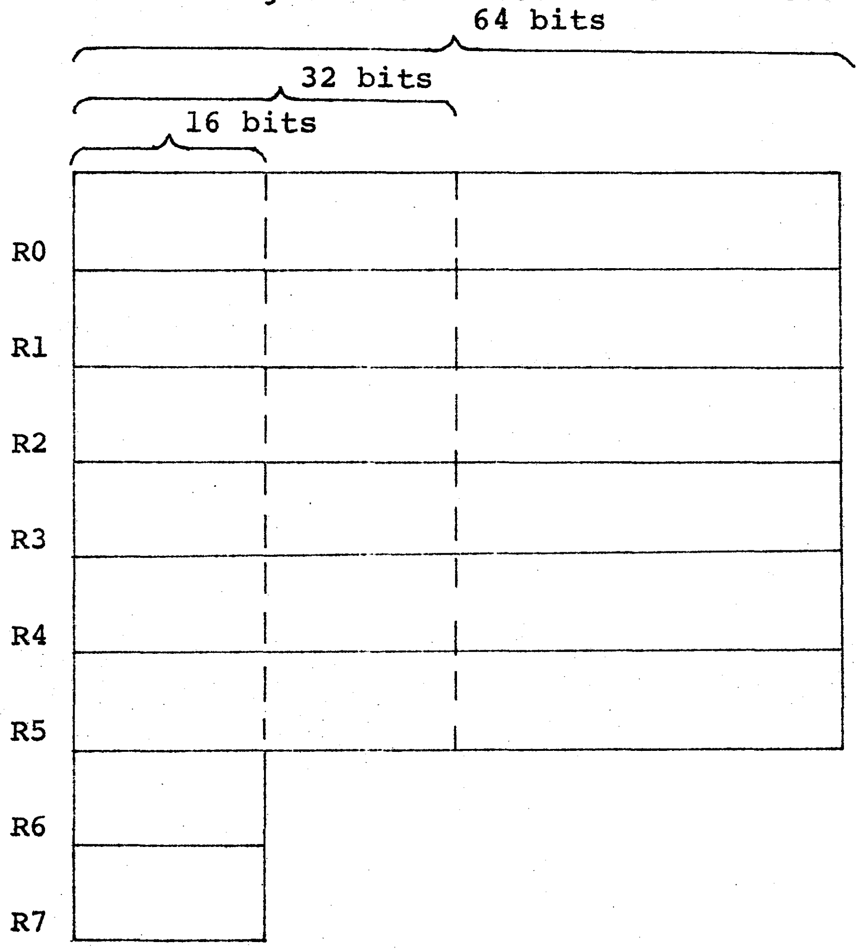


Figure 1-1. Variable Length Register Layout

When a register is used for 16-bit integer arithmetic, the top 16 bits are used; when a register is used for 32-bit integer/floating arithmetic the top 32 bits are used, etc.

The disadvantage of this scheme is that the total number of general purpose "GP" registers is still 8 like in the PDP-11/20 while the number of data types has increased.

1.2 The Completely Overlapping Scheme

The register layout for this scheme is shown in Figure 1-2. It consists of 8 double registers D0-D7 which are 32 bits wide. The bottom 4 double registers D4-D7 overlap the current PDP-11/20 registers R0-R7.

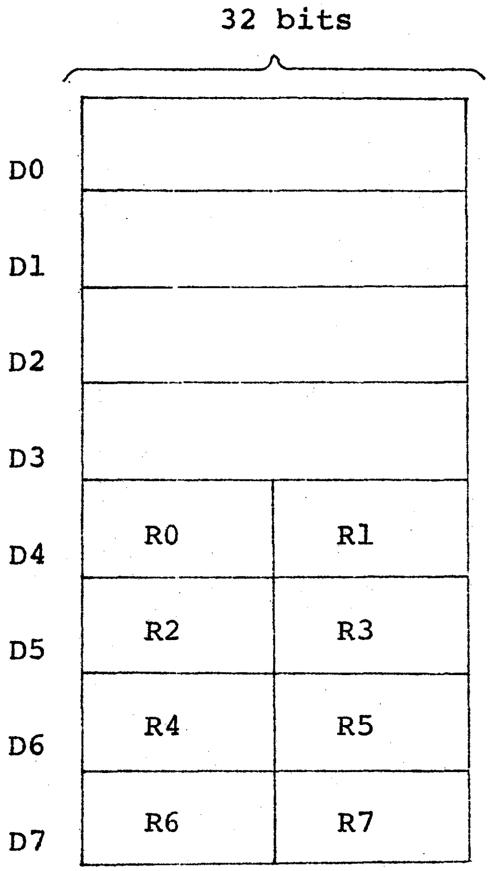


Figure 1-2. Completely Overlapping Register Layout

Advantages of this scheme:

1. Four extra 32-bit or two extra 64-bit registers are obtained.

2. The registers D4-D7 overlap R0-R7.

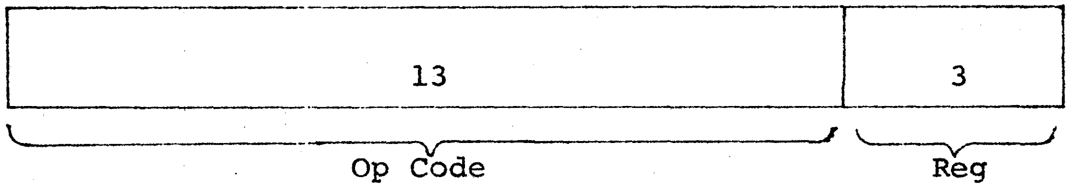
   A. Floating and 32-bit integer quantities can be loaded into the registers R0-R7 which allows for easier software simulation of floating-point and 32-bit integer arithmetic.

How is SP & PC saved ??
R6 & R7 are sacred.

B. In subroutines frequently registers (i.e., 16-bit registers) have to be saved and restored. The overlap allows this to be done in groups of 2 and 4 registers.

Disadvantages of this scheme:

Some problems arise in the use of the CONVERT instructions convert between the following data types: 16-bit and 32-bit integer and 32-bit and 64-bit floating point. The instruction format is shown below:



Convert Instruction Format

Because of the lack of sufficient op code space, only one address field (the 3-bit Reg field) could be included in the register. The Reg field determines both: the source register as well as the destination register. When the data is a 16-bit integer, Reg denoted R0-R7 else Reg denotes D0-D7.

Some examples:

CDF 3 means: 32-bit integer in D3 is converted into a 32-bit floating point number in D3.

CID 3 means: 16-bit integer in R3 is converted into a 32-bit integer in D3.

It is quite clear that with this scheme CID4 and CID5 are very dangerous instructions because the results get stored into D4 and D5 respectively which wipes out R0, R1 and R2, R3 respectively. This means that in compilers R4 and R5 are very tricky to use.

1.3 The Partially Overlapping Scheme

The register layout for this scheme is shown in Figure 1-3. It has 7 double registers D0-D5 and D7 and 8 single registers R0-R7. D7 overlaps R0, R1.

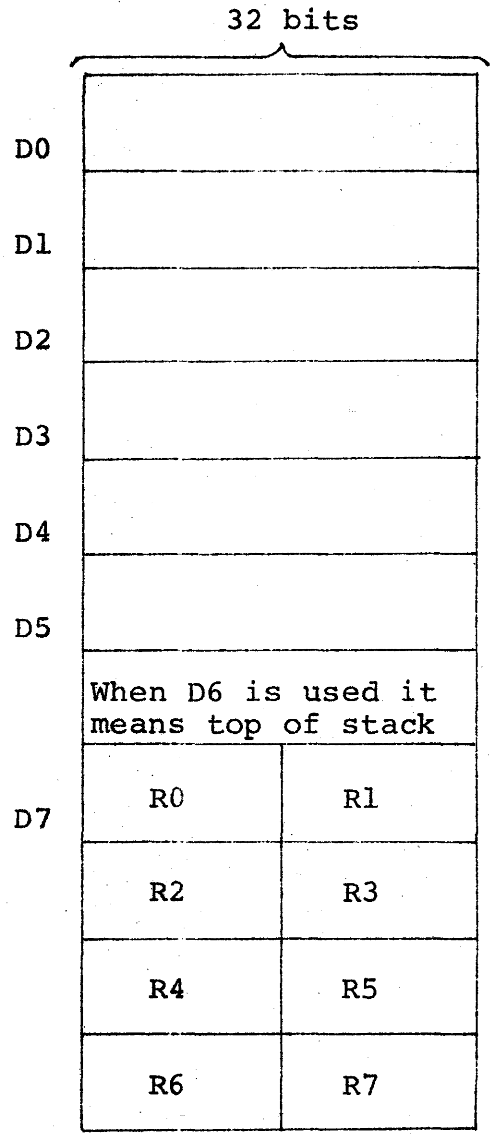


Figure 1-3. Partially Overlapping Register Layout

This scheme has two advantages above that of section 1.2 "The Completely Overlapping Scheme."

1. Two more (non-overlapping) double registers are available, i.e., more GP registers.

2. The problem of the CONVERT instruction is solved.

Disadvantages of this scheme:

Hardware implementation imposes some problems because the internal addressing of a register cannot be done by a simple masking of the register bits from the instruction. An addition has to take place, i.e., 14 (decimal) has to be added to R as specified in the instruction to find its corresponding scratchpad location, see Figure 1-4.

D0	S0	S1
D1	S2	S3
D2	S4	S5
D3	S6	S7
D4	S8	S9
D5	S10	S11
	S12	S13
D7	R0 S14	R1 S15
	R2 S16	R3 S17
	R4 S~~17~~ 18	R5 S~~18~~ 19
	R6 S~~19~~ 20	R7 S~~20~~ 21

Note: S # are the numbers of the scratchpad registers.

Figure 1-4. Scratchpad Register Layout for the Partially Overlapping Scheme

1.4 The Disjoint Scheme

The register layout for this scheme is shown in Figure 1-5. It has 6 double registers D0-D5 and 8 single registers R0-R7 which do not overlap.

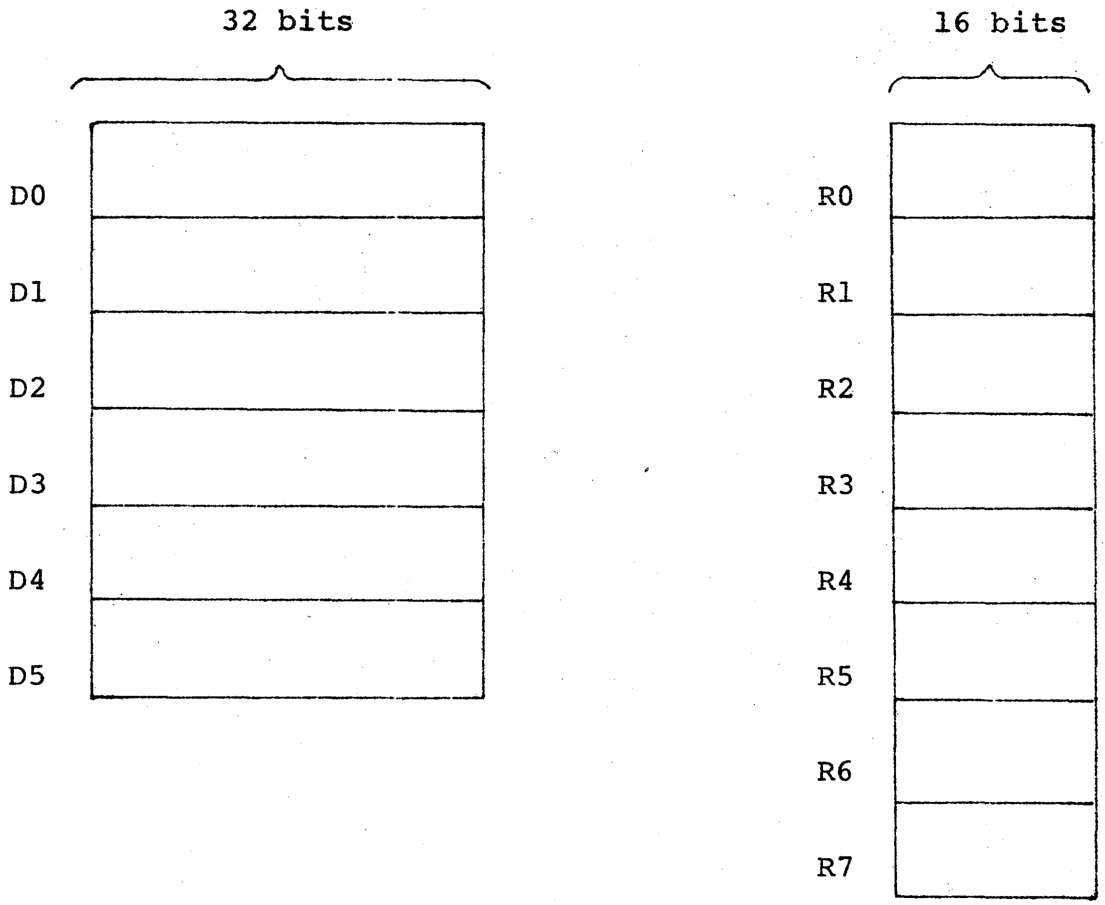


Figure 1-5. Disjoint Register Layout Scheme

The advantages are the same as for solution 1.3 "The Partially Overlapping Scheme" except for the absence of any overlap between the D and R registers. This, however, can be compensated for by laying out the registers over core memory.

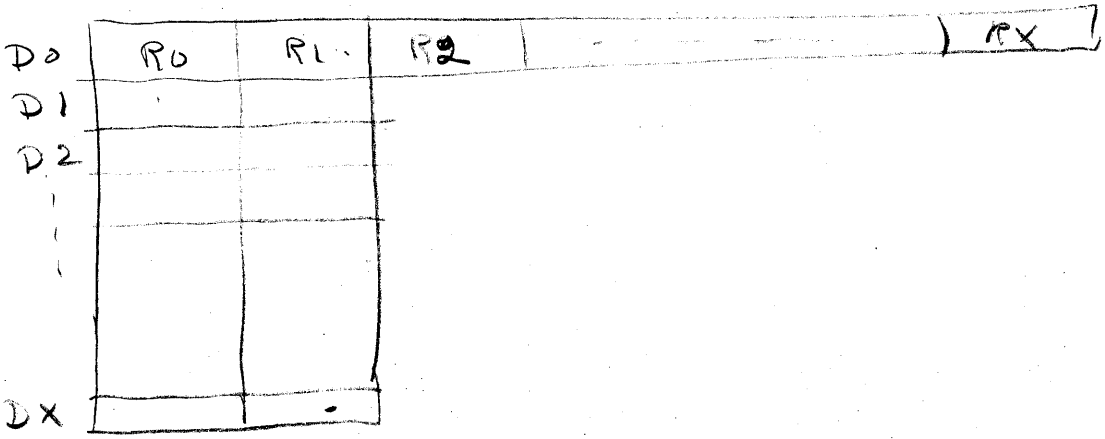
Comparing this register organization with that of the IBM 360 family, it should be noted that the 360 family has 4 floating point registers which can be used as 32 or 64-bit registers.

disadvantage — lose of unique I/O address space.

The 11/40, assuming this register format, has six 32-bit floating-point registers or three 64-bit floating-point registers.

Because of the advantages of this scheme and the lack of most of the disadvantages, this scheme will be considered for implementation.

Consider — Jim Bell


D0
D1
D2
DX
R0
R1
R2
RX