```
+---------------------------+
|   |   |   |   |   |   |   |
| d | i | g | i | t | a | l |
|   |   |   |   |   |   |   |
+---------------------------+
```

interoffice memorandum

To: Monitor Groups
Layered Products Groups

Date: 27 Apr 83
From: Mike Uhler
Dept: Jupiter Engineering
DTN: (8-)231-6448
Loc/Mail stop: MR01-2/E85
Net mail: UHLER at 10

Subject: Programming for a Pipelined Machine

1.0 Introduction

The KC10 processor is the first implementation of the PDP-10 architecture to include significant amounts of pipelined logic. Pipelining techniques have increased the performance of the machine, but at the same time they have caused problems in obtaining a smooth instruction flow through the EBOX.

Because pipelining implies that multiple instructions are being prefetched or executed at one instant in time, problems of instruction interaction come up. For example, if the instruction currently being executed in the EBOX stores into a location that has been previously prefetched by the IBOX, a conflict exists between the correct data stored by the EBOX and the prefetched data contained in the IBOX.

This memo points out ways that a programmer can minimize the impact of instruction interactions by carefully aranging the instruction sequence. These suggestions apply, not only to assembly language programmers, but also to compiler code generators. I have made an attempt to distinuish those suggestions that apply to pipelined machines in general and those that apply specifically to the KC10 processor.

The memo contains one section for each major type of interaction. Each section contains a definition of the problem, examples that demonstrate the problem, and suggestions for minimizing it. A final section give a priority to each of the interactions as it applies to the KC10 processor.

2.0 Pipelining

As PDP-10 processors begin to use pipelining techniques, the interactions between instructions in the instruction stream begin to affect the performance of the machine. In the past there has been little or no interaction between instructions; execution was done as a strictly sequential fetch-and-execute scheme.

The KL10 was the first processor to take the step into pipelining, and that step was only through simple instruction prefetch. In a very limited number of cases, the KL10 overlapped the instruction fetch of the next instruction with the end of execution of the current instruction. All other aspects of execution including EA-calc and operand fetch were done strictly sequentially.

The KC10 processor extends the prefetch capability considerably through the use of an independent IBOX to prefetch instructions and operands for EBOX execution. With the current design, up to three instructions can be in the process of being prefetched or executed at one time. Future PDP-10 processor may add considerably more pipelining to achieve high agregate instruction performance.

The very thing that increases the performance of a pipelined machine also causes problems.

3.0 AC conflicts

3.1 Conflicts on the AC of a jump instruction

3.2 Conflicts on the AC of a skip instruction

3.3 Conflicts on the AC of other instructions (AC forwarding)

4.0 Memory operand conflicts

4.1 Conflicts on the memory operand of skip instructions

4.2 Conflicts on the memory operand of other instructions

4.3 Conflicts on ACs used as memory operands

5.0 EA conflicts

5.1 XR conflicts