An Overview of PDP-8 Architecture

(09/08/2003)

Memory

32 "pages" of 128 12-bit words = 4K memory

12 bit addressing (bits 0 - 4 is page, bits 5 - 11 is offset)

```
 <    p a g e      > <       o f f s e t       >
|___|___|___|___|___|___|___|___|___|___|___|___|
  0   1   2   3   4   5   6   7   8   9   10  11
The bits in a word are numbered left to right from 0 to 11
```

To convert 12-bit octal addresses to page/offset format, convert to binary and "regroup" . For example

```
        3167o = 011 001 110 111 = 01 100 | 1 110 111 = page 14o offset 167o
```

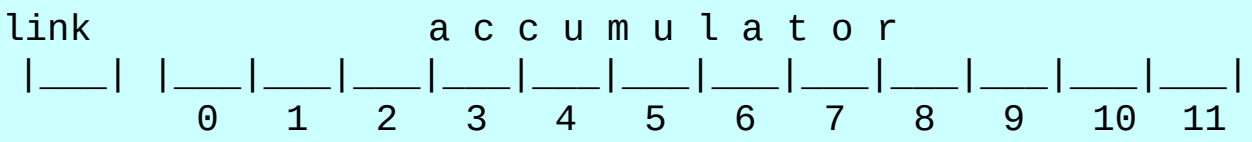
Reading and Writing Memory

Access to memory is controlled by two registers: a 12-bit Central Processor Memory Address register (CPMA) for the address and a 12-bit Memory Buffer register (MB) for the value.

To read memory a 12-bit address is loaded into the CPMA and a read signal is asserted. The contents of memory appears in the MB; i.e. MB <- Mem(CPMA). To write memory the address and value to be written are loaded into the CPMA andMB registers respectively and a write signal is asserted; i.e. Mem (CPMA) <- MB.

CPU Registers

- 12 bit Accumulator (AC) + Link Bit (carry out)



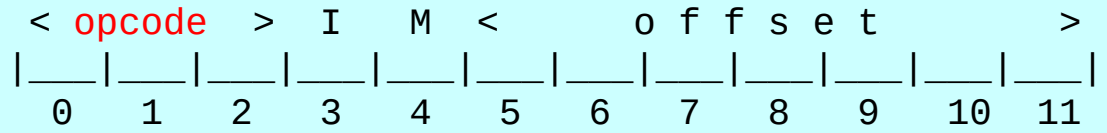
- 12 bit Multiplier Quotient (MQ) register
- 12 bit Program Counter (PC) holds address of next instruction
- 3 bit Instruction Register (IR) holds the op-code of current instruction
- 12 bit MB (Memory Buffer)
- 12 bit CPMA (Central Processor Memory Address register)
- 12 bit Console Switch Register (SR) - accessible from console switches & used to load memory from console

---

Instruction Formats

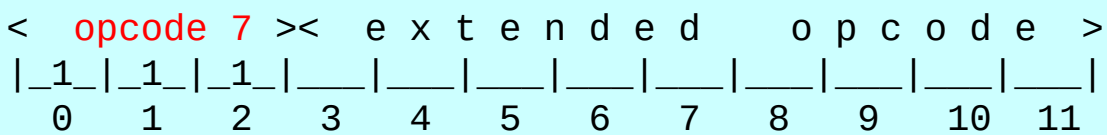
one address; "Load and Store" architecture; eight 3-bit op-codes

- MRI (Memory Reference Instruction) Format; opcodes 0 - 5

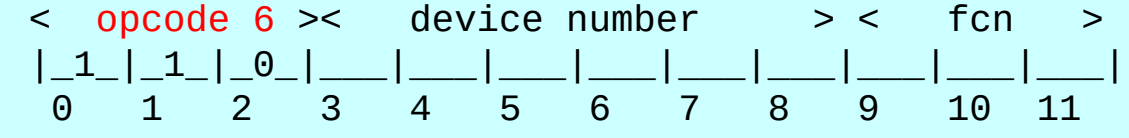



```
bit 3 (Indirect bit)    = 0 No Indirection; = 1 Indirection
bit 4 (Memory Page bit) = 0 Zero Page;      = 1 Current Page
```

- Operate (opcode 7) - instructions that operate on accumulator and link



- Test I/O (opcode 6) - input/output instructions

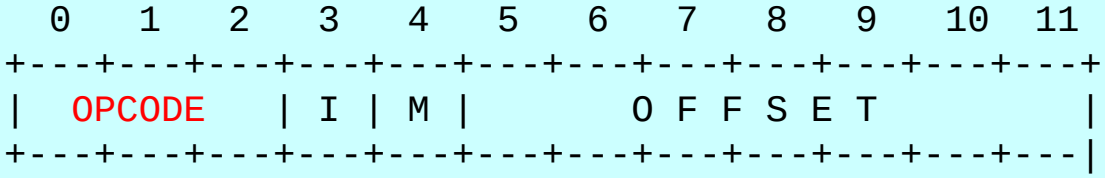


Addressing Modes : Effective Address Calculation

With three bits (bits 0 - 2) allocated to the op-code, the PDP-8's fixed length instruction format did not have enough bits left over for a 12-bit address. Instead a memory reference instruction contained a 7 bit field (bits 5 - 11) for a page offset. 12 bit addresses were calculated by pre-fixing the 7-bit page offset with a 5-bit page number. The page number was either page zero or the same page as the address of the instruction (called the current page - the memory reference being restricted to the same page as the instruction). Bit 4 was used to determine which. Since only two pages (256 words) could be directly addressed, bit 3 was used for indirect addressing. In all there were four ways to calculate an effective address, the last of which, auto-indexing, was used to implement arrays and strings.

- Zero Page - Access to addresses 0000o - 0177o
  - bit 4 = 0
  - Effective Address := 00000 + offset
- Current Page - Access to addresses on same page as instruction
  - bit 4 = 1
  - Effective Address := Instruction Page + offset
- Indirect Addressing - access to any address but costs two memory reads
  - bit 3 = 1
  - Effective Address = C(00000 or Instruction Page + offset)
- Autoindexing - used to implement arrays
  - indirection through addresses 0010o - 0017o
  - C(0010o - 0017o) = C(0010o - 0017o) + 1 Effective Addresses:= C(0010o - 0017o)

MRI Instructions - these instructions accessed memory




```
Bits 0 - 2     : Operation Code
Bit 3          : Indirect Addressing Bit (0:Direct/1:Indirect)
Bit 4          : Memory Page (0:Zero Page/1:Current Page)
Bits 5 - 11    : Offset Address
```

- C( ) denotes "contents of"
- EAddr denotes "effective address"

```
 0 - AND   AND accumulator with memory
```

```
                C(AC) <- C(AC) and C(EAddr)
   1 - TAD   Two's complement Add to accumulator
                C(AC) <- C(AC) + C(EAddr)
                If carry out then complement Link
   2 - ISZ   Increment memory and Skip if Zero
                C(EAddr) <- C(EAddr) + 1
                If C(EAddr) = 0 then C(PC) <- C(PC) + 1
   3 - DCA   Deposit Accumulator to memory and Clear accumulator
                C(EAddr) <- C(AC);  C(AC) <- 0
   4 - JMS   JuMP to Subroutine
                C(EAddr) <- C(PC) ;  C(PC) <- EAddr + 1
   5 - JMP   JuMP
                C(PC) <- EAddr
```

Some Useful Opcode 7 Instructions - While there are over 50 opcode 7 instructions, the 11 below are perhaps the most useful

```
   7040 - Complement Accumulator (CMA)
   7001 - Increment Accumulator (IAC)
   7041 - Complement and Increment Accumulator (negate AC) (CIA)
   7300 - Clear AC and Link (CLA CLL)
   7402 - Halt (HLT)
   7500 - Skip next instruction if Accumulator is negative (SMA)
   7440 - Skip next instruction if Accumulator is zero (SZA)
   7510 - Skip next instruction if Accumulator is positive (SPA)
   7450 - Skip next instruction if Accumulator not equal to zero
(SNA)
   7010 - Rotate Accumulator link pair Right (RAR)
   7004 - Rotate Accumulator link pair Left (RAL)
```

Calculating Effective Addresses for PDP-8 MRI Instructions: It is useful to be able to able to convert 12-bit octal addresses into page/offset notation and to obtain the 12-bit effective address from a MRI instruction.

- Converting between page offset (page (bits 0 - 4) /offset (bits 5 - 11)) and 12-bit octal addresses

```
     6224o = 110 010 010 100 = 11001  0010100 = page 31o offset
024o
```

- address/contents notation is used to show the contents of memory at a given address

```
      6224/1367 is address 6224 contents 1367
      (or 1367 is found at address 6224)
```

- The following three line diagram is used to determine the Effective Address of MRI instruction Given 6224/1367 find the Effective Address (Bit 3 = 0 : No Indirection / Bit 4 = 0 : Current Page)

```
                                                  012 345 678 901
<- bits
6224o = 110 010 010 100 <- address/contents -> 001 011 110 111 =
1367o
           |                                                |
        110 01 (page)                                 1 110 111
(offset)
           |
        110 011 110 111 = 6367o <- Effective Address
```

- Given 6224/1607 find the Effective Address (Bit 3 = 1 : Indirection / Bit 4 = 1 : Current Page)

```
                                                  012 345 678 901
<- bits
6224o = 110 010 010 100 <- address/contents -> 001 110 000 111 =
1607o
           |                                                |
        110 01 (page)                                 0 000 111
(offset)
           |
        110 010 000 111 = 6207o <- Address of Effective Address
      So if we have 6207/3521 then 3521 is the Effective Address
```

Input Output Test Instructions : Opcode 6 : Opcode 6 instructions did input/output with bits 3 - 8 indicating the device and bits 9 - 11 indicating the specific operation.

```
  0   1   2   3   4   5   6   7   8   9   10  11
+---+---+---+---+---+---+---+---+---+---+---+---+
| 1 | 1 | 0 |  device number        | opcode    |
+---+---+---+---+---+---+---+---+---+---+---+---|
```

```
Bits 0 - 2     : Op Code 6
Bits 3 - 8     : Device Number
Bits 9 - 11    : Extended Opcode (operation specification bits)
```

Operate Instructions : Opcode 7 - Opcode 7 instructions operate on the accumulator link pair (which is why they do not need to reference memory). There are actually three different groups of opcode 7 instructions and within each group individual operations are controlled a single bits. These *micro-operations* (when practiable) may be executed in parallel. For example, CLA (Clear Accumulator) and CLL (Clear Link) may be executed at the same time.

Group 1 Microinstructions : Bit 3 = 0 : These operations are used to clear, complement, rotate and/or increment the accumulator-link pair.

```
  0   1   2   3   4   5   6   7   8   9   10  11
+---+---+---+---+---+---+---+---+---+---+---+---+
| 1 | 1 | 1 | 0 |CLA|CLL|CMA|CML|RAR|RAL|0/1|IAC|
+---+---+---+---+---+---+---+---+---+---+---+---|
```

```
Bit 10 : Rotate 1 if 0; Rotate 2 if 1
```

Group 2 Micro Instructions : Bit 3 = 1 and Bit 11 = 0 : These operations are used for conditional branching (skip on condition), for example SMA (bit 5) = Skip on Minus Accumulator or SZA (bit 6) = Skip on Zero Accumulator

```
  0   1   2   3   4   5   6   7   8   9   10  11
+---+---+---+---+---+---+---+---+---+---+---+---+
| 1 | 1 | 1 | 1 |CLA|SMA|SZA|SNL|0/1|OSR|HLT| 0 |
+---+---+---+---+---+---+---+---+---+---+---+---|
```

```
Bit 8 : if 1 reverse logic to obtain SPA, SNA, SZL
```

Group 3 Microinstructions : Bit 3 = 1 and Bit 11 = 1 : These operations work with the MQ register.

```
  0   1   2   3   4   5   6   7   8   9   10  11
+---+---+---+---+---+---+---+---+---+---+---+---+
| 1 | 1 | 1 | 1 |CLA|MQA| 0 |MQL| 0 | 0 | 0 | 1 |
+---+---+---+---+---+---+---+---+---+---+---+---|
```

Return to Comp 255 Home Page
