Addressing Modes

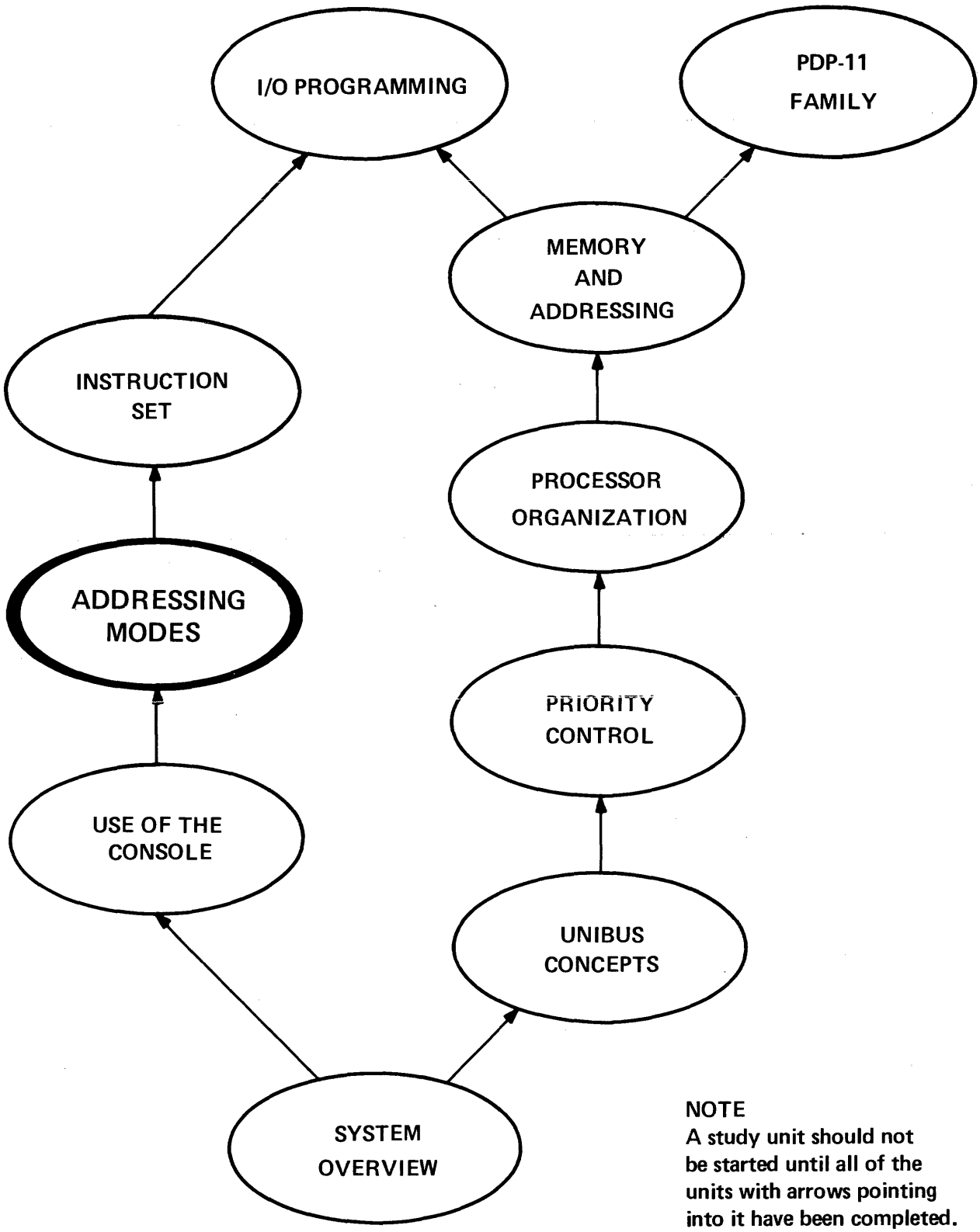
student workbook introduction to the pdp11

digital equipment corporation · maynard. massachusetts

1st Printing, March 1974
2nd Printing (Rev), November 1974
3rd Printing (Rev), April 1977

Printed in U.S.A.

course map



NOTE
A study unit should not be started until all of the units with arrows pointing into it have been completed.

read on

READ LEARNING OBJECTIVES (page 1)

NOW RUN FILM CARTRIDGES A & B

REVIEW MATERIAL (pages 3 – 9) AND COMPLETE EXERCISE A (page 15)

NOW RUN FILM CARTRIDGE C

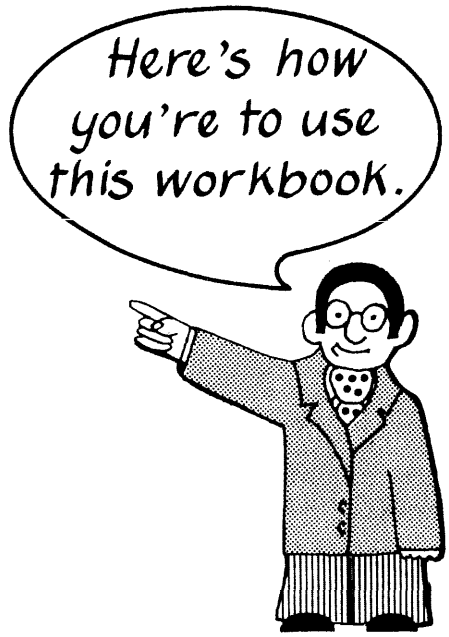
REVIEW MATERIAL (pages 10 – 14) AND COMPLETE EXERCISE B (page 19)

TAKE TEST & CHECK RESULTS (pages 21 – 23)

ANY INCORRECT ANSWERS

YES → PLEASE REVIEW THE MATERIAL YOU'RE HAVING DIFFICULTY WITH. (ADDITIONAL RESOURCES ARE LISTED ON PAGE 2.)

NO → GOOD WORK! NOW GO ON TO THE NEXT STUDY UNIT.



read on

objectives

After completing this study unit, you should be able to . . . .

- ★ Explain how each of the eight basic addressing modes are used to locate operands stored in a PDP-11 system.
- ★ Point out the similarities and differences between the eight basic addressing modes and cite examples where each mode would be used.
- ★ Describe the difference between direct and deferred addressing.
- ★ Explain and use the four special addressing modes involving the program counter (PC).
- ★ Recognize and use the assembler syntax and octal code for all the addressing modes.
- ★ Define terms such as "base," "index," "pointer," "effective address," "offset," and "position-independent code" as they relate to the addressing modes.
- ★ Explain the difference between relative addressing and absolute addressing.

- **PDP-11/04/05/10/35/40/45 Processor Handbook**

Read Chapter 3, Addressing Modes. (Paragraph 3.7 contains an excellent summary of the different addressing modes.)

Also, read Chapter 5, Paragraph 5.5 (Position-Independent Code).

review material
film cartridges A & B

The following material is covered in this study unit:

Topic	*Key Points*	*Visual Ref.*
GPRs	★ All addressing within the PDP-11 system is accomplished by way of the general-purpose registers. • Register 7 also functions as the program counter. It's automatically incremented by two each time the CPU fetches another instruction word in the program. • To fetch the next instruction from memory, the CPU takes the instruction's address stored in the PC and places it on the Unibus.	3–7
basic instruction format	★ After the instruction is retrieved, it is decoded by the CPU. • Part of the decoded instruction tells the CPU what operation to perform. This is called the "op code" or operation code. • The remainder of the instruction tells the CPU how to locate the value (or operand) that it is to operate upon. This part of the instruction further breaks down into an *addressing mode* and a *register* field.	8–10
register field	★ Three bits are required to select any one of the eight GPRs. They specify which GPR is to be used with the current instruction.	11
addressing modes	★ Three more bits specify *how* the selected GPR is to be used in order to locate the operand. This is called the "addressing mode." There are 8 different ways of using the GPRs and, therefore, 8 basic addressing modes.	12
addressing modes: examples	★ Here are 3 examples showing how the GPRs can be used.	13–15

read on ➧

Topic	*Key Points*	*Visual Ref.*
	• We can store operands right in the GPRs.	
	• We can store an address in a GPR which directs the CPU to an operand located in memory.	
	• We can store a pointer in a GPR. This pointer directs the CPU to a memory location containing the address of an operand stored in an I/O register.	
8 basic modes	★ By combining 8 basic addressing modes with any of our general-purpose registers, we can locate operands stored anywhere in the PDP-11 system.	16–17
	• Now let's examine each of the 8 basic modes and see how they are used to locate operands.	
mode 0	★ Register mode: *operand* is stored in a GPR.	18–22
	• Provides quickest access to data; there's no need to tie-up the bus to retrieve the operand.	
	• Instruction operates directly on the contents of the GPR.	
	• General-purpose registers are labelled R0 through R7 (R6=SP, R7=PC).	
mode 0: symbol	★ Assembler notation for mode 0: INC R3. This instruction tells the CPU to increment the value contained in register 3.	20, 21
mode 1	★ Register deferred mode: *address* of the operand is stored in a GPR.	23–28
	• Address contained in the GPR directs the CPU to the operand. Operand is located outside the CPU – either in memory or an I/O register.	
	• Retrieval of the operand involves a DATI or DATIP bus cycle.	

read on ➧

Topic	*Key Points*	*Visual Ref.*
mode 1: symbol	★ Assembler notation for mode 1: DEC (R0) or DEC @ R0. Either symbol tells the CPU that register 0 contains the address of the operand that is to be decremented.	27
mode 2	★ Autoincrement mode: GPR contains the *address* of the operand; address is *automatically incremented* after the operand is retrieved. Address now references the next sequential operand. • Allows a programmer to automatically step through a list or series of operands stored in *consecutive* locations. • When an instruction calls for mode 2, the address stored in the GPR is autoincremented each time the instruction is executed. • Address is autoincremented by 1 if we are working with bytes, or by 2 if we are working with words.	29–39
mode 2: symbol	★ Assembler notation: CLRB (R4)+. Register 4 contains the address of a byte which is to be cleared; then the address is incremented by 1 so it points to the next byte in the list.	39
mode 4	★ Autodecrement mode: GPR contains an *address* that is *automatically decremented;* then the decremented address is used to locate an operand. • Similar to autoincrement, but allows a programmer to step through a list of words or bytes in the reverse order. • Address is autodecremented by 1 for bytes and by 2 for words.	42–45
mode 4: symbol	★ Assembler notation: CLR–(R4). First, decrement the address stored in register 4. Then clear the word location specified by the decremented address.	48

read on ➧

Topic	*Key Points*	*Visual Ref.*
autoincr. vs autodecr.	★ Both addressing modes automatically modify an address stored in a GPR. Each time the address is incremented or decremented, it advances us to the next operand in the list. • Autoincrement *first* uses the address to locate an operand, *then* it *increments* the address. • Autodecrement *first* decrements the address, *then* it uses the new address to locate an operand.	46–48
mode 3	★ Autoincrement deferred mode: GPR contains a pointer to an address that is stored outside the CPU. When the address is retrieved, the GPR's pointer is automatically incremented *by two.*	52–61
mode 3: example	★ One way of accessing operands stored in I/O registers is to place a table of I/O addresses in memory. Each entry in the table addresses a different *set* of I/O registers. • To get to this table of I/O addresses, the table's starting address is placed in a GPR. Thus, the GPR "points" to the table. • To step through this table of I/O addresses, mode 3 is used to automatically increment the pointer stored in the GPR. • Thus, mode 3 provides for automatic stepping through a table of addresses as a means of accessing operands.	52–58
mode 3: symbol	★ Assembler notation: CLR @ (R3)+. • The "@" denotes this is a deferred addressing mode; i.e. R3 contains a pointer to an address. • The "+" indicates the pointer in R3 is incremented *by two after* the address is located.	60

read on ➧

Topic	*Key Points*	*Visual Ref.*
mode 3 vs mode 2	★ Mode 2 (autoincrement) is only used to access operands that are stored in *consecutive* locations. Mode 3 (autoincrement deferred) is used to access lists of operands stored anywhere in the system; i.e., the operands do not have to reside in adjoining locations. Mode 2 is used to step through a table of *values*; mode 3 is used to step through a table of *addresses.*	62–65
mode 5	★ Autodecrement deferred mode: GPR contains a pointer. The pointer is *first* decremented by two, *then* the new pointer is used to retrieve an address stored outside the CPU. • Similar to autoincrement deferred, but allows a programmer to step through a table of addresses in the reverse order. • Each address then redirects the CPU to an operand. Note that the operands do not have to reside in consecutive locations.	67–72
mode 5: symbol	★ Assembler notation: CLR @ – (R0). • The "@" denotes this is a deferred addressing mode; i.e. R0 contains a pointer. • The "–" signifies an autodecrement function. It is placed in front of (R0) because the pointer is decremented by two before it is used for retrieving an address.	73
mode 6	★ Index mode: a *base* address is summed with an *index* word to produce the effective address of an operand. The *base* address specifies the starting location of a table or list. The *index* word then represents the address of an entry in the table or list *relative* to the starting (base) address. • The *base* address may be stored in a GPR. In this case, the *index* word follows the current instruction. Or, the locations of the *base* address and *index* word may be reversed (*index* word in GPR; *base* address following the current instruction).	77–101

read on ➧

Topic	*Key Points*	*Visual Ref.*
mode 6: example # 1	★ Suppose we have a table of operands. • To get to the table, we use the table's starting address. This starting address is *fixed* and can serve as our base address. • To get to *any* entry in the table, we use a displacement from the starting (base) address. This *variable* displacement serves as our *index* word. It defines the location of any entry relative to the base address. • To retrieve any entry from the table, we use mode 6 to *sum* the *base* with the appropriate *index* word. The *base* can be stored in a GPR and the *index* word in the first memory location following the current instruction (or the locations of the *base* and *index* word can be *reversed*).	78–91
mode 6: example # 2	★ Suppose we are working with several tables and wish to update the second entry in each table. • We use a starting address to define the location of each table. Since the starting address is different for each table, it now becomes the *variable* component. This starting address serves as our *index* word. • We are interested in the second entry in all tables. The displacement of this entry is *fixed* – it's always plus two. Because the displacement is fixed, it serves as our *base.* • To locate the second entry in any table, we again use mode 6 to sum the *base* of plus two with the appropriate starting address or *index* word. The base can be stored in a GPR. The index word then follows the current instruction. Or, we could reverse the locations of these two address components.	92–101

read on ➧

Topic	*Key Points*	*Visual Ref.*
mode 6: symbol	★ Assembler notation: CLR ± X (R2). • "X" represents the index word (or base address) that is stored in the *first* location following the clear instruction. • Register 2 contains a base address (or index word) that is summed with "X" to produce the effective address of the operand.	85
mode 7	★ Index deferred mode: a base address is summed with an index word. The result is a *pointer* to an address – rather than the actual address. • Similar to mode 6, except that it produces a pointer to an address. The address, in turn, redirects the CPU to the desired operand. • Provides for the random access of operands using a table of operand addresses.	103–106
mode 7: symbol	★ Assembler notation: INC @ ± X (R1). • The "@" indicates this is a deferred addressing mode. • "X" represents the index word (or base) that is stored in the first location following the increment instruction. • Register 1 contains a value that is summed with "X." Since this is a deferred mode, the result is a pointer instead of the actual address.	106

NOTE 1

Visuals 108–116, and the accompanying narration, summarize the eight basic addressing modes.

NOTE 2

A review of the special PC addressing modes covered in film cartridge C begins on the next page.

Topic	*Key Points*	*Visual Ref.*
special PC modes	★ There are 4 special addressing modes involving just the program counter. • The PC is unique from the other GPR's in one important respect. Whenever the processor retrieves an instruction, it automatically advances the PC by 2. • By combining this automatic advancement of the PC with four of the *basic* addressing modes, we produce the 4 special PC modes – immediate, absolute, relative, and relative deferred.	119–122
immediate mode	★ This special mode is equivalent to the autoincrement (mode 2) using the PC. In this case, the operand immediately follows the instruction. • The PC is incremented twice when used with addressing mode two. • First, it's automatically updated by 2 when the instruction is retrieved. The new value in the PC (PC+2) is the address of the operand; i.e. the operand follows the instruction. • Then, the PC is *autoincremented* by 2. The new value, PC+4, addresses the next instruction in the program.	123–128
immediate mode: symbol	★ Assembler notation: INC #100. • This instruction tells the CPU to increment the value 100. • The "#" designates the immediate addressing mode. Therefore, the value to be incremented (100) immediately follows the instruction word. The value 100 is actually part (a second word) of the increment instruction.	128

read on ▶

Topic	*Key Points*	*Visual Ref.*
absolute mode	★ This special mode is equivalent to the autoincrement deferred mode using the PC. In this case, the *address* of the operand immediately follows the instruction. • The PC is incremented twice during the absolute mode. • First, it's automatically updated by 2 when the instruction is retrieved. The new value, PC+2, points to a memory location containing the *address* of the operand; i.e. the address follows the instruction. • Then, the PC is *autoincremented* by 2. The new value, PC+4, addresses the next instruction in the program.	130–140
absolute mode: symbol	★ Assembler notation: INC @ #1000. • This instruction tells the CPU to increment the operand stored in memory location 1000. • The "@" denotes this is a deferred mode. Therefore, the PC points to the address 1000. This address is actually the second word of this 2-word instruction.	137, 138
"absolute" address	★ When the absolute addressing mode is used, the location of the operand remains fixed no matter where the instruction is located in memory. • The operand's address is constant (absolute); therefore, this is called the *absolute* addressing mode.	139, 140

read on ➧

Topic	*Key Points*	*Visual Ref.*
relative mode	★ This special mode is the same as index mode 6 except that it uses the PC. The operand's address is calculated by adding the word that follows the instruction (called an "offset") to the updated contents of the PC. • PC+2 directs the CPU to the offset that follows the instruction. • PC+4 is summed with this offset to produce the effective address of the operand. PC+4 also represents the address of the next instruction in the program.	141–151
"relative" address	★ With the relative addressing mode, the address of the operand is always determined with respect to the updated PC. Therefore, when the instruction is relocated, the operand is moved by the same amount.	150
offset	★ The distance between the updated PC and the operand is called an "offset." When a program is assembled, this offset appears in the first word location that follows the instruction.	150
relative mode: symbol	★ Assembler notation: INC A. • "A" is a label that identifies the location of the operand to be incremented.	151
relative deferred mode	★ This special mode is the same as index deferred (mode 7) except that it uses the PC. A *pointer* to an operand's address is calculated by adding an offset (that follows the instruction) to the updated PC. • This mode is similar to the relative mode, except that it involves one additional level of addressing to obtain the operand. • The sum of the offset and updated PC (PC+4) serves as a pointer to an address. When the address is retrieved, it can be used to locate the operand.	153–158

read on ➧

Topic	*Key Points*	*Visual Ref.*
relative deferred: symbol	★ Assembler notation: INC @ A. • The "@" denotes a deferred addressing mode. • "A" is a label that identifies the location that contains the *address* of the operand.	158

NOTE
Visuals 159–167, and the accompanying narration, summarize the four special PC addressing modes.

RELATIVE vs ABSOLUTE ADDRESSING & POSITION INDEPENDENT CODE

The description that follows should be read carefully. (This information is not included in the sound-filmstrip cartridges.)

position independent code (PIC)

★ On the PDP-11, it is possible to write program code which, although originally designed to run in one set of memory locations, will work equally well if the code is relocated in another area of memory. Such code is called "position-independent code" or PIC.

- Programs written in PIC do *not* directly reference any *absolute* locations within the moving code. Instead, all such references are given *relative to the PC*; that is, the locations are specified in terms of offsets from the current value of the PC.

- Thus, if an instruction and the location it is addressing are moved (relocated), and the *relative* distance between them is not changed, the *same offset* (relative to the PC) can be used.

read on ➧

problem: accessing locations

★ One problem in writing PIC is how to correctly access various locations. Two situations occur:

1. The location to be accessed is *not* part of the moving code, or

2. The location to be accessed is *within* the moving code.

location not in moving code

★ If the location we wish to access is *not* in the moving code, we use *absolute* addressing.

Example: INC @#TKS, where TKS = 177 560 (the absolute address of the Teletype® keyboard status register).

location within moving code

★ In the example below, the location labelled "SAVE" that we wish to address is *within* the moving code and moves with it. Therefore, the *relative* mode is used (mode 6 indexed on the PC).

```
PC                INC SAVE

PC+2                   (offset to SAVE)

PC+4              BR AWAY

PC+6      SAVE:   0
```

Wherever these instructions are loaded, the offset (2) is added to the updated PC (PC+4). The result (PC+6) is the effective address of location SAVE. Thus, the INCREMENT instruction always adds one to the contents of SAVE.

basic addressing modes

Complete the following chart. This is an instruction list, not a program. Assume the initial conditions apply for each instruction.

NOTE TO STUDENT

If you find it difficult to complete this exercise, review the material on pages 3–9 of this workbook and work through the examples contained in Chapter 3 of the PDP-11 Processor Handbook. Then return to this workbook exercise.

INITIAL CONDITIONS:

(R0) = 1000	(1000) = 100	(100) = 10	
(R2) = 3000	(3000) = 300	(300) = 30	
(R5) = 4000	(4000) = 400	(400) = 40	
(R7) = 7000	(6504) = 654	(654) = 64	
(2776) = 276	(276) = 26	(26) = 6	
(3500) = 350	(350) = 30	(30) = 0	
(4100) = 410	(410) = 40	(40) = 0	
(2777) = 177			

NOTE

The contents of a register or memory location are symbolized by enclosing the address in parentheses. For example, (100)=10 means location 100 contains 10.

INSTRUCTION	MODE OCTAL CODE	EFFECTIVE ADDRESS *	CONTENT OF SELECTED GPR AFTER USE	OPERAND BEFORE EXECUTION	OPERAND AFTER EXECUTION	NEW PC
INC @ R5	1	4000	4000	400	401	7002
DEC (R2) +						
CLR @ -(R2)						
CLR R5						
INC 100 (R5)						
DECB @ (R2) +						
CLRB (R5)						
CLRB - (R2)						
INC @ 1000 (R2)						
CLRB (R0) +						
INC - 300 (R5)						
INC @ - 300 (R5)						

*Effective Address is the final address after all address decoding has been completed.

read on

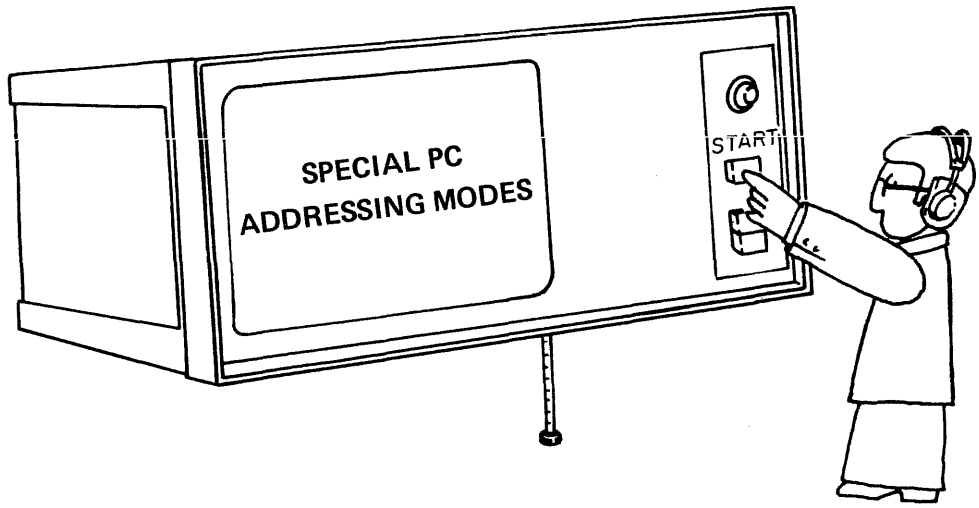
basic addressing modes

INITIAL CONDITIONS:

(R0) = 1000	(1000) = 100	(100) = 10
(R2) = 3000	(3000) = 300	(300) = 30
(R5) = 4000	(4000) = 400	(400) = 40
(R7) = 7000	(6504) = 654	(654) = 64
(2776) = 276	(276) = 26	(26) = 6
(3500) = 350	(350) = 30	(30) = 0
(4100) = 410	(410) = 40	(40) = 0
(2777) = 177		

INSTRUCTION	MODE OCTAL CODE	EFFECTIVE ADDRESS	CONTENT OF SELECTED GPR AFTER USE	OPERAND		NEW PC
				BEFORE EXECUTION	AFTER EXECUTION	
INC @ R5	1	4000	4000	400	401	7002
DEC (R2) +	2	3000	3002	300	277	7002
CLR @ - (R2)	5	276	2776	26	00	7002
CLR R5	0	R5	0000	4000	0000	7002
INC 100 (R5)	6	4100	4000	410	411	7004
DECB @ (R2) +	3	300	3002	30	27	7002
CLRB (R5)	1	4000	4000	400	400	7002
CLRB - (R2)	4	2777	2777	177	000	7002
INC @ 1000 (R2)	7	400	3000	40	41	7004
CLRB (R0) +	2	1000	1001	100	000	7002
INC - 300 (R5)	6	3500	4000	350	351	7004
INC @ - 300 (R5)	7	350	4000	30	31	7004

read on



RETURN TO A/V
FILM CARTRIDGE C

special pc addressing modes

Complete the following chart. Assume the initial conditions apply for each instruction.

INITIAL CONDITIONS:
(R7) = 5000
(1000) = 100, and (100) = 10, and (10) = 1
(SUM) = 2700; assume SUM is offset $+600_8$ relative to PC.
(LIMIT) = 4000; assume LIMIT is offset -500_8 relative to PC.
(7500) = 1000

INSTRUCTION	MODE OCTAL CODE	2nd WORD OF INSTRUCTION	EFFECTIVE ADDRESS	OPERAND	
				BEFORE EXECUTION	AFTER EXECUTION
CLR #1234	27	001234	5002	1234	all 0's
DEC @ #1000					
CLR @ 7500					
CLR 7500					
INC @ #100					
CLR 1000					
INC #1234					
INC @ 1000					
INC @ - 4004 (R7)					
INC SUM					
CLR LIMIT					

read on

special pc addressing modes

INITIAL CONDITIONS:
(R7) = 5000
(1000) = 100, and (100) = 10, and (10) = 1
(SUM) = 2700; assume SUM is offset $+600_8$ relative to PC.
(LIMIT) = 4000; assume LIMIT is offset -500_8 relative to PC.
(7500) = 1000

	INSTRUCTION	MODE OCTAL CODE	2nd WORD OF INSTRUCTION	EFFECTIVE ADDRESS	OPERAND BEFORE EXECUTION	OPERAND AFTER EXECUTION
	CLR #1234	27	001234	5002	1234	all 0's
	DEC @ #1000	37	001000	1000	100	77
	CLR @ 7500	77	002474	1000	100	all 0's
*	CLR 7500	67	002474	7500	1000	all 0's
	INC @ #100	37	000100	100	10	11
	CLR 1000	67	173774	1000	100	all 0's
	INC #1234	27	001234	5002	1234	1235
**	INC @ 1000	77	173774	100	10	11
	INC @ - 4004 (R7)	77	173774	100	10	11
	INC SUM	67	000600	5604	2700	2701
	CLR LIMIT	67	177300	4304	4000	all 0's

* The instruction CLR 7500 assembles as follows:

(5000) = CLR 7500
(5002) = Offset
(5004) = Next Instruction

Offset = Effective Address–Updated PC
Offset = $7500_8 - 5004_8 = 2474_8$

** The instruction INC @ 1000 assembles as follows:

(5000) = INC @1000
(5002) = Offset
(5004) = Next Instruction

Offset = Address of Effective Address–Updated PC
Offset = 1000–5004
Offset = 173774 (2's complement since result is negative)

read on

When you have completed the study unit, please take this self-scoring test. Then compare your answers against the "answer sheet" which can be obtained from your supervisor. Based on your test results, either review the appropriate material in this study unit or proceed to the next unit in the series.

1. Match the following statements with the corresponding addressing mode.

   a. Operand address is computed as an offset to the updated PC.

   b. Contents of R3 are summed with a index word to produce an effective address.

   c. No bus cycle is required to retrieve operand.

   d. Operand follows the instruction.

   e. Base plus index word produces a pointer to the operand's address.

   f. Automatic advancement through a list of items stored in *consecutive* locations.

   g. Fixed address of operand follows the instruction.

   h. Directs the CPU to a series of operands via a table of addresses stored in memory.

   i. Address of operand is stored in a GPR; contents of GPR are not modified.

   ( ) Register – mode 0
   ( ) Register deferred – mode 1
   ( ) Autoincrement – mode 2
   ( ) Autoincrement deferred – mode 3
   ( ) Index – mode 6
   ( ) Index deferred – mode 7
   ( ) Immediate – mode 2 with PC
   ( ) Absolute – mode 3 with PC
   ( ) Relative – mode 6 with PC

read on ➧

2. Complete the following chart. Assume the initial conditions apply for each instruction.

INITIAL CONDITIONS:

		(R5) = 1777
(R1) = 700	(R4) = 2000	(R7) = 4000
(1000) = 100	(100) = 10	(10) = 1
(1776) = 176	(176) = 16	(16) = 6
(1777) = 177	(177) = 17	(17) = 7
(2000) = 200	(200) = 20	(20) = 2
(7000) = 700	(700) = 70	(70) = 7

(TABLE) = 7000; assume TABLE is offset -1700_8 relative to the PC.

INSTRUCTION	MODE OCTAL CODE	2nd WORD OF INSTRUCTION	EFFECTIVE ADDRESS	OPERAND BEFORE EXECUTION
INC 7000	67	002774	7000	700
DEC @ 1100 (R1)				
DEC #60				
CLRB (R5) +				
DEC @–(R4)				
INC @ 1776				
INC - 600 (R1)				
CLR R4				
CLR TABLE				
CLRB -(R4)				
INC @ #7000				
CLR @ R4				

read on

3. Given:

DEC @ 50 (R2)	with	(R2) = 350
		(420) = 220
		(220) = 100
		(400) = 300

a. The operand is stored in location _____ .

b. After the instruction is executed, the contents of that location are _____ .

notes