

ORG test 2 13 Nov 85

1-4 10 pts each 5-8 3 15 pts each

Good Luck!

- A 1010
- B 1011
- C 1100
- D 1101
- E 1110
- F 1111

0011 1100
 1111
 1011 1101
 1100 0100

 10000001

1. In an 8-bit machine $3C_{16}$ is subtracted from BD_{16} . Give the contents of the flags C, S, V, Z and PE (parity even) after this operation.

C = 1 S = 1 V = 0 Z = 0 PE = 0

2. AB. Formulate a mapping Process from (macro) op-codes to (μ) control memory addresses given that μ -memory has 4K words, op-codes are 6 bits wide and 4 control words are needed for each macro-op

00XX XXX00

C. Illustrate your map with the op-code 42_{16} , give your address in binary

00 101010 00

D. What is the size of the CAR? 10

3. AB. The text divides all interrupts into 3 classes. Give these 3 classes and an example of each.

- ext (asyc) I/O input
- int (trap) $\div 0$
- software (syscall) output

C. What is an interrupt vector and how is it used?

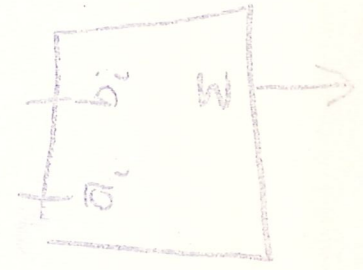
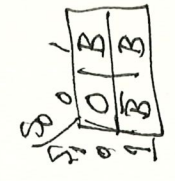
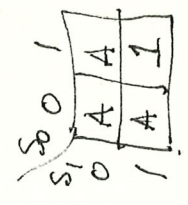
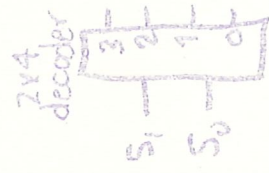
an address in memory associated with interrupt either loc of interrupt routine or pointer to same

4. Design a circuit for ALU which yields

(Actually we need only one "stage" of the logic which is "fed" into a parallel adder.) The decoder will ease the pain

S ₀	C _{in} = 0	C _{in} = 1
00	A	A+1
01	A+B	A+B+1
10	A+B	A+B+1
11	B-1	B

A_i | B_i



1. The instruction is fetched from memory.

2. The instruction is decoded to determine the operation and the operands.

3. The instruction is executed.

4. The instruction is written back to memory.

5. The instruction is fetched from memory.

6. The instruction is decoded to determine the operation and the operands.

7. The instruction is executed.

8. The instruction is written back to memory.

9. The instruction is fetched from memory.

10. The instruction is decoded to determine the operation and the operands.

11. The instruction is executed.

12. The instruction is written back to memory.

13. The instruction is fetched from memory.

14. The instruction is decoded to determine the operation and the operands.

15. The instruction is executed.

16. The instruction is written back to memory.

17. The instruction is fetched from memory.

18. The instruction is decoded to determine the operation and the operands.

19. The instruction is executed.

20. The instruction is written back to memory.

21. The instruction is fetched from memory.

22. The instruction is decoded to determine the operation and the operands.

23. The instruction is executed.

24. The instruction is written back to memory.

25. The instruction is fetched from memory.

26. The instruction is decoded to determine the operation and the operands.

27. The instruction is executed.

28. The instruction is written back to memory.

ORG 36

NOP I CALL INDIRECT

DECSF, PCTBR, BRTPC U JMP NEXT

SPTAR U JMP NEXT

WRITE U JMP FETCH#

ORG 48

SPTAR U JMP NEXT

MUL SP, READ U JMP NEXT

BRTPC U JMP FETCH

(Addressing modes) The 8080 has 8 addressing modes. It is a 16-bit machine with four 8-bit registers (Accumulator, PC, SP, and SR). The general format for an instruction is:



opcode 7 bits for 'mode' or immediate 'copy'. The 'mode' bit is 0 for 'mode' and 1 for 'immediate'.

The reg. field is 4 bits for 'reg. reference'. The reg. field is 0 for 'reg. direct', 1 for 'reg. indirect', 2 for 'index', 3 for 'pre-decrement', 4 for 'post-decrement'.

(For 8080) 0 immediate, 1 direct, 2 indirect, 3 relative, 4 pre-decrement, 5 post-decrement, 6 pre-increment, 7 post-increment.

Mode 0: Immediate. Mode 1: Direct. Mode 2: Indirect. Mode 3: Relative. Mode 4: Pre-decrement. Mode 5: Post-decrement. Mode 6: Pre-increment. Mode 7: Post-increment.

$$R3 \leftarrow X$$

$$M[X] \leftarrow M[R2]$$

$$M[R4-1] \leftarrow M[PC+X] \quad \& \quad R4 \leftarrow R4-1$$

$$M[M[X]] \leftarrow M[R0] \quad \& \quad R0 \leftarrow R0+1$$

$$M[R3+Y] \leftarrow M[R2+X]$$

(A-ops)

7.1 This version of the chapter 5 computer has both an SP (points to TOS, grows toward low memory) and an FP (a "frame pointer"). The LINK ass-instruction reserves some

stack space (for subroutine variables). The amount of space is given by the address field (it's a memory reference instruction). It does PUSH FP, FP ← SP, SP ← SP + constant. UNLINK undoes this (but it doesn't know the "constant" (nor does it need it)). Write the execution cycles for these with controls given they have "op-codes" 1 & 5 respectively. (Additional addresses are available)

A. LINK

```

c2t0: SP ← SP - 1, MBR ← FP
t1: MAR ← SP, FP ← SP
t2: M ← MBR, SP ← SP + constant
t3: change stacks
SP ← SP - 1

```

B. UNLINK

```

c2t0: SP ← FP, MAR ← FP
t1: MBR ← M, SP ← SP + 1
t2: FP ← MBR
t3: change stacks

```

```

MAR ← SP, MBR ← FP, FP ← SP, SP ← SP + MBR(AD)
M ← MBR

```

8. (parameter passing via the stack) [SP points to TOS, grows toward low memory]

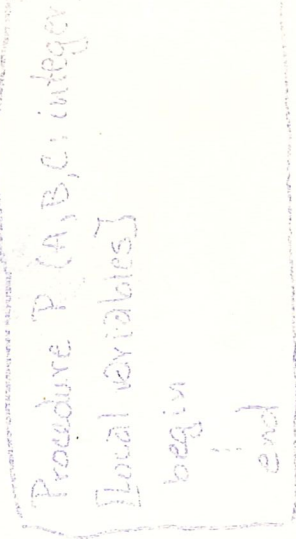
A compiler would generate the code

```

push 10
push 4
push 1
call P

```

for a procedure call P(1,4,10) if the procedure body is as in the box.



A. Suppose integers and addresses fit into 1 word of memory. At the beginning of P the variables A,B,C are easy to obtain via using the SP as an index register. What are the offsets from SP for Var A? +1 Var B? +2 Var C? +3

B. If Procedure P has local variables it will call LINK (Prob 7) to create stack space for them. The variables A,B,C are still easy to obtain but this time via indexing from the FP. What are the offsets from FP for Var A? +2 Var B? +3 Var C? +4

CDE. This stack use allows for procedures to be called with a different number of parameters each time it is called. i.e. push the variables onto the stack in backwards order (right to left) then push the size of the variables onto the stack (i.e. reals take 2 words) then push the number of variables onto the stack (i.e. reals and call the routine. Like writeln (integer, real, integer) and call the routine. What is the offset of the first variable from SP in general?

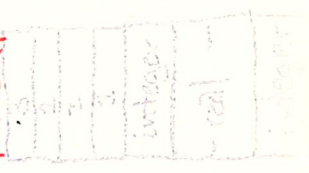
$M[SP] + 1 + 1$

D. What is the offset of the second variable from SP?

$M[SP] + M[SP + 1] + 1 + 1$

E. What is the offset of the third variable from SP?

$M[SP] + M[SP + 1] + M[SP + 2] + 1 + 1 + 1$



OR6 test 2 13 Nov 85

1-4 10 pts each 5-8 15 pts each Good Luck!

1. In an 8-bit machine $3C_{16}$ is subtracted from BD_{16} . Give the contents of the flags C, S, V, Z and PE (parity even) after this operation.

$$5/0/7/0/4/0/1/0/0/1/0$$

$$\begin{array}{r} 50 \\ 56 \\ 24 \\ 4 \\ 1 \\ \hline 135 \end{array} \quad \begin{array}{r} 135 \\ 180 \\ \hline \approx 75.0\% \end{array}$$

2. AB. Formulate a mapping process from (macro) op-codes to μ control memory addresses given that μ -memory has 1K words, op-codes are 6 bits wide and 4 control words are needed for each macro-op

$$14/0/1/0/1/0/1/0/1/0/0$$

$$\begin{array}{r} 140 \\ 8 \\ 6 \\ 4 \\ \hline 88.9\% \end{array}$$

C. Illustrate your map with the op-codes, give your address in binary.

D. What is the size of the CAR?

3. AB. The text divides all interrupts into 3 classes. Give those 3 classes and an example of each.

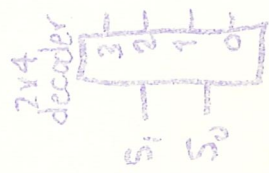
$$\begin{array}{r} 100 \\ 18 \\ 8 \\ 14 \\ 12 \\ 4 \\ \hline 10/2/1/2/2/0/1/0 \rightarrow \end{array}$$

$$86.7\%$$

C. What is an interrupt vector and how is it used?

4. Design a circuit for ALU which yields (Actually we need only one "stage" of the logic which is "fed" into a parallel adder.) The decoder will ease the pain

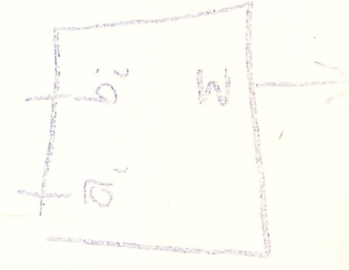
S ₁ S ₀	C _{in} =0	C _{in} =1
00	A	A+1
01	A+B	A+B+1
10	A+B	A+B+1
11	B-1	B



$$8/2/2/0/2/0/0/0/3/0/1/1$$

$$\begin{array}{r} 80 \\ 18 \\ 16 \\ 12 \\ 6 \\ 0 \\ \hline 132 \end{array}$$

$$73.3\%$$



2. (a) - (b) The number of...
 means the...
 links to...
 in the...
 of...
 these...
 effective...

3. (a) - (b) The number of...

2/0/0/2/0/3/0/1/4/1/2/0/1/2/0/0

$$\frac{180}{90}$$

30
 24
 30
 28
 6
 10
 3
 4

 143

$$\frac{143}{270} \approx 53.0\%$$

6. (Address: ...)
 with...
 The...

100 = ...
 19 = ...

of... is the...

means...
 the...
 the...
 (see...)
 1. ...
 2. ...
 3. ...
 4. ...
 5. ...

15
 14
 24
 22
 30
 8
 7
 6
 5
 16

 149

1/1/0/2/2/3/0/1/1/1/1/1/1/1/1/1/0/0/0

55.2%

