Computer Organisation & Architecture Home Assignment Solution

Sem: 4TH **Section: IT7, IT8** **Faculty: Prof. Anil Kumar Swain**

DOA: Dt.03.02.2020 **DOS:Dt.14.02.2020**

Roll Number		Full Signature	

Instruction: Answer all questions.

1. What is Stored Program Concept?

Answer:

All modern computers use the stored program concept which was initially conceived by the design team of ISA computer led by Von Neumann. Hence, it is commonly known as **Von Neumann** concept. The essentials of stored program concept are as follows:

- The computer has five different types of units: memory, ALU, control unit, input unit and output unit.
- The program and data are stored in a common memory.
- Once a program is in memory, the computer can execute it automatically without manual intervention.
- The control unit fetches and executes the instructions in sequence one by one. This sequential execution can be modified by certain type of instructions.
- An instruction can modify the contents of any location in memory. Hence, a program can modify itself; instruction-execution sequence also can be modified.

N.B: A computer with a **von Neumann architecture** stores program data and instruction data in the same memory; a computer with a **Harvard architecture** has separate memories for storing program and data.

2. What are the functionalities of MAR, MDR, IR, PC, SP and GPRs?

Answer:

Refer page no.8 or 412 of CO Book by Carl Hammacher, 5th ed.

3. What is the difference between word addressable and byte addressable memory?

Answer:

Byte Addressable Memory	Word Addressable Memory
a) Byte addressable memory refers to memory address that is accessed one byte at a time. i.e. Here byte is fixed that is 8 bit always.	a) Word addressable memory refers to memory address that is accessed one word at a time. ie. Here word is not fixed that typically range from 16 to 64 bits.
b) Individual byte has a unique address.	b) Individual Word has a unique address.

Computer Organisation & Architecture Home Assignment Solution

Sem: 4TH Section: IT7, IT8 Faculty: Prof. Anil Kumar Swain

DOA: Dt.03.02.2020 DOS:Dt.14.02.2020

4. What is difference between Big Endian and Little Endian byte order.

Answer:

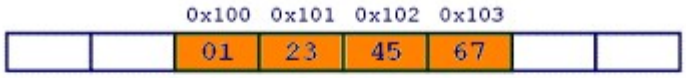
- Little and big endian are two ways of storing multibyte data-types (int, float, etc).
- In byte ordering, the "big end" byte is called the "high-order byte" or the "most significant byte".
- The term 'endian' as derived from 'end' may lead to confusion. **The end denotes which end of the number comes first** rather than which part comes at the end of the sequence of bytes. The basic endian layout can be seen in the table below:

Endianness	First Byte	Last Byte
Big	Most Significant	Least Significant
Little	Least Significant	Most Significant

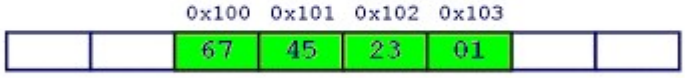
Big Endian Byte Order	Little Endian Byte Order
a) The most significant byte (the "big end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next bytes in memory. **Example**: Macintosh Computer/Machine	a) The least significant byte (the "little end") of the data is placed at the byte with the lowest address. The rest of the data is placed in order in the next bytes in memory. **Example:** Intel 80x86 Computer/Machine

Example: Represent the integer ox01234567 (represented in hexa) in Big Endian and Littel Endian machines.

Answer:



Big Endian



Little Endian

5. If a computer a runs a program in 10 seconds and B runs the same program in 15 seconds, how much faster is A than B?

Answer:

We know that A is n times faster than B if

$$\frac{\text{Performance of A}}{\text{Performance of B}} = \frac{\text{Execution Time for B}}{\text{Execution Time for A}} = n$$

=>15/10=1.5

A is therefore 1.5 times faster than B.

6. **To maximize performance, we want to minimize response time or execution time for some task (True/ False).**
Answer:
True

7. **Do the following changes to a computer system increase throughput, decrease response time or both?**
Case1: Replacing the processor in a computer with a faster version.
Case2: Adding additional processors to a system that uses multiple processors for separate tasks. For example searching the world wide web.
Answer:
In case1: both response time and throughput are improved because decreasing response time almost always improves throughput. In case2: Only throughput increases because no one task gets work done faster.

8. **A processor is advertised as having a speed of 2GHz. What 2GHz refers. Explain.**
Answer:
2GHz referes to clock rate. This means that it can process data internally 2 billion times a second (every clock cycle). If the processor is a 32-bit processor running at 2GHz then it can potentially process 32 bits of data simultaneously, 2 billion times a second !!

9. **Derive Basic Performance Equation.**
Answer:
- We know, Response time or Execution time is the time taken by the computer , to execute a given program, from start to end of the program.
- There are **three** equally important components of execution time.
  i) **Clock cycle time/clock period** – It is just the length of a cycle.
  ii) **CPI -** It is is the average number of clock cycles per instruction, for a particular machine and program.
  iii) **Number of instructions in a program** – It is the **dynamic instruction count** that is how many instructions are actually executed when the program runs, not the static instruction count that is how many lines of code are in a program.

- The basic performance equation, which is fundamental to measuring computer performance, measures the CPU time, is as follows.
  - ✓ CPU Time To execute a program = T= time/program
  - ✓ Time require to execute a basic step = P = time/cycle
  - ✓ Average number of basic steps required to execute one machine instruction = CPI = S = cycles/instruction
  - ✓ Number of instructions a program contains is N = instructions/program
  Now

$$\frac{\text{Time}}{\text{Program}} = \frac{\text{Time}}{\text{Cycle}} \times \frac{\text{Cycles}}{\text{Instruction}} \times \frac{\text{Instructions}}{\text{Program}}$$

$$\text{CPU Time} = P \times \text{CPI} \times N$$

$$T = P \times S \times N$$

$$T = \frac{S \times N}{R}$$

Reference: CO/ Clarl Hamacher Book, 5th Ed/ Page-15 →

$$\boxed{T = \frac{N \times S}{R}}$$

Remember

10. A processor advertised as having a 900 MHz clock is always provides better performance than a 700-MHz processor (True/False). Justify your answer.

Answer:

False because both processors may have a different value of S.

11. When run on a given system, a program takes 1,000, 000 cycles. If the system achieves a CPI of 40, how many instructions were executed in running the program?

Answer:

We know CPI = Cycles/Instructions

⇨ Instructions = Cycles/CPI = 1,000, 000/ 40 = 25, 000

12. Given a binary pattern in some memory location, is it possible to tell whether this pattern represents a machine instruction or a number?

Answer:

No; any binary pattern can be interpreted as a number or as an instruction.

13. At the end of a memory read operation, the MDR is loaded with a binary combination, how that combination is interpreted as an instruction or an operand to an instruction?

Answer:

If the memory operation is initiated by sending the contents of PC to MAR, then the content of MDR will be interpreted as an instruction else as an operand.

14. The content of register R1 is 11010110. What will be the decimal value after execution of the following instructions. Assume the number is represented in 2's complement format.

a) RotateL #2, R1

b) RotateL #3, R1

Answer:

a) RotateL #2, R1

After rotation R1's content will be 01011011

In Decimal the value = $2^6 + 2^4 + 2^3 + 2^1 + 2^0 = 64 + 16 + 8 + 2 + 1 = 91$

b) RotateL #3, R1

After rotation R1's content will be 10110110
The number is -ve, so
In Decimal the value = -(2's of 10110110)=
= -(01001010) =-($2^6 + 2^3 + 2^1$) = -(64 + 8 + 2) = -74

15. The content of register R1 is 10010110. What will be the decimal value after execution of the follwoing individual instructions. Assume the number is represented in 2's complement format.

a) LShiftL #2, R1
b) AShiftR +3, R1

Answer:

a) **LShiftL #2, R1**

After logical left shift of R1 by 2 bits, R1's content will be 01011000
In Decimal the value = $2^6 + 2^4 + 2^3 = 64 + 16 + 8 = 88$

b) **AShiftR #3, R1**

After aithmatic right shift of R1 by 3 bits, R1's content will be 11110010
The number is -ve, so
In Decimal the value = -(2's of 11110010)
= -(00001110)=-(8 + 4 + 2) = -14

16. The content of register R2 (24 bit) is 10011000 10110011 11010111.Write the instruction for performing the following operations:

a) Clear the LSB of R2 to 0.
b) Clear the MSB of R2 to 0.
c) Clear the 2nd byte of R2 to 0.
d) Set the MSB of R2 to 1.
e) Set LSB of R2 to 1.
f) Clear all bits of R2

Answer:

The binary number **10011000 10110011 11010111 in hexa is 98 B3 D7**

a) Clear the LSB of R2 to 0

And #$FFFF00, R2

b) Clear the MSB of R2 to 0

And #$00FFFF, R2

c) Clear the 2nd byte of R2 to 0

And #$FF00FF, R2

d) Set the MSB of R2 to 1

Or #$FF0000, R2

e) Set LSB of R2 to 1

Or #$0000FF, R2

f) Clear all bits of R2

Clear R2

17. A memory byte location contains the pattern 01000001. What does this pattern represent when interpreted as a binary number? What does it represent as an ASCII code?

Answer:

The decimal equivalent of the binary 00101100 = $2^6 + 2^0 = 64 + 1 = 65$. The pattern represents 65 and the ASCII character 'A' it represents.

Note: ASCII stands for **American Standard Code for Information Interchange.** It is a code that uses seven bits for representing 128 English characters as numbers, with each letter assigned a number from 0 to 127. Computers can only understand numbers, so an ASCII code is the numerical representation of a character such as 'a' or '@' or an action of some sort (Enter key, Esc key etc).

Extended ASCII uses eight instead of seven bits, which adds 128 additional characters. This gives extended ASCII the ability for extra characters, such as special symbols, foreign language letters, and drawing characters etc.

18. An instruction is a 24 bit instruction. It is a byte addressable memory. The PC contains 300. Which one of the following is a legal PC value:

(a) 400 (b) 500 (c) 600 (d) 700

Answer:

(c) 600

19. A processor has 40 distinct instructions and 24 general purpose registers. A 32-bit instruction word has an opcode, two register operands and an immediate operand. The number of bits available for the immediate operand field is _____.

Answer:

16

6 bits are needed for 40 distinct instructions (because, 32 < 40 < 64)

5 bits are needed for 24 general purpose registers(because, 16< 24 < 32) 32-bit instruction word has an opcode(6 bit), two register operands(total 10 bits) and an immediate operand (x bits). The number of bits available for the immediate operand field => x = 32 -

Addressing Modes and Assemble Languages

20. Write an assembly language program to derive the expression X = A*B+C*D

a) in a stack based computer with zero address instructions
b) In a acumulator based CPU with one address instuctions
c) In register based CPU with two address instructions
d) In register based CPU with three address instructions

Answer:

X = A*B+C*D
(with zero address instructions)

Step-1: Convert the expression X= A*B+C*D into postfix (RPN) form

X = A*B+C*D
= (AB*)+C*D
=(AB*)+(CD*)
= AB*CD*+ (Make the RPN bracket free)
1 2 34 5 67

Step-2 : Write the assembly language

Now Taking symbols from left to right, if symbol is a operand write instruction to push the operand into the stack. If the symbol is an operator write the appropriate opcode. For + operator write ADD, for * operator write MULT etc.

Assembly language

```
1. PUSH   A     ; TOS ← [A]
2. PUSH   B     ; TOS ← [B]
3. MULT         ; TOS ← [A] * [B]
4. PUSH   C     ; TOS ← [C]
5. PUSH   D     ; TOS ← [D]
6. MULT         ; TOS ← [C] * [D]
7. ADD          ; TOS ← [A] * [B]  + [C] * [D]
8. POP    X     ; X ←[TOS]
```

X = A*B+C*D
(with one address instructions)

Assembly language	RTN
LOAD A	; AC ← [A]
MULT B	; AC ← [AC] * [B]
STORE T	; T ← [AC]
LOAD C	; AC ← [C]
MULT D	; AC ← [AC] * [D]
ADD T	; AC ← [AC] + [T]
STORE X	; X ← [AC]

X = A*B+C*D
(with two address instructions)

Assembly language	RTN
MOVE A, R1	; R1 ← [A]
MULT B, R1	; R1 ← [R1] * [B]
MOVE C, R2	; R2 ← [C]
MULT D, R2	; R2 ← [R2] * [D]
ADD R1, R2	; R2 ← [R2] + [R1]
MOVE R2, X	; X ← [R2]

N.B: Always try to store the result in a register.
If two operands are allowed as memory operands then we rewrite the above as follows

Assembly language	RTN
MULT A, B	; B ← [B] * [A]
MULT C, D	; D ← [D] * [C]
ADD B, D	; D ← [D] * [B]
MOVE D, X	; X ← [D]

X = A*B+C*D
(with three address instructions)

Assembly language	RTN
MULT A, B, R1	; R1 ← [A] * [B]
MULT C, D, R2	; R2 ← [C] * [D]
ADD R1, R2, X	; X ← [R1] + [R2]

21. Registers R1 and R2 of a computer contain the decimal values 1200 and 4600. What is the effective address of the memory operand in each of the following instructions?

a) Load 20(R1), R5
b) Move #3000, R5
c) Store R5, 30(R1,R2)
d) Add −(R2), R5
e) Subtract (R1)+, R5

[CO/Hamacher Book/5th Ed/Exercise/2.13/Page-100]

Answer:

a) 1220 (**Explanation:** R1=1200, 20(R1)=> **EA**=[R1] + 20 = 1200 + 20 =1220)
b) Part of the instruction
c) 5830 (**Explanation:** R1=1200, R2=4600, 30(R1, R2)=> **EA**=[R1] +[R2] + 30 = 1200 + 4600 + 30 = 5830)
d) 4599 (**Explanation:** R2=4600, -(R2) => R2 = [R2] - 1 = 4600 -1 =4599, **EA** = [R2] = 4599)
e) 1200 (**Explanation:** R1=1200, (R2)+ => **EA** = [R2] = 1200, R2 = [R2] + 1 = 1201)

N. **B.** If each memory word consumes 4 bytes and machine is byte addressable, then answer of
d) 4596 (**Explanation:** R2=4600, -(R2) => R2 = [R2] - 4 = 4600 -4 =4596, **EA** = [R2] = 4596
e) 1200 (**Explanation:** R1=1200, (R2)+ =>**EA** = [R2] = 1200, R2 = [R2] + 4 = 1204)

22. Both of the following statements cause the value 300 to be stored in location 1000, but at different times.

ORIGIN 1000
DATAWORD 300
and
Move #300,1000

Explain the difference.

[CO/Hamacher Book/5th Ed/Exercise/2.16/Page-100]

Answer:
The assembler directives **ORIGIN and DATAWORD** cause the object pro gram memory image constructed by the assembler to indicate that 300 is to be placed at memory word location 1000 at the time the program is loaded into memory prior to execution.
The **Move** instruction places 300 into memory word location 1000 when the instruction is executed as part of a program.

Stack & Subroutine

23. Suppose that a stack runs from location 2000 (BOTTOM) downto no further than location 1500 (TOP) and each stack word consumes 4 bytes and machine is byte addressable. The stack pointer is loaded initially with the address value 2004. Write a routine for both PUSH and POP operation.

Answer:

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPUSH (If it allows auto increment/ decrement addressing mode)	Compare Branch ≤ 0 Move	#1500, SP FULLERROR NEWITEM, -(SP)	Check to see if the stack pointer contains an address value equals to or less than 1500. If it does, the stack is full. Branch to the routine FULLERROR for appropriate action. Otherwise, decrement the stack pointer and push the element in memory location NEWITEM onto the stack.

Or

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPUSH (If it does not allow auto increment/ decrement addressing mode)	Compare Branch ≤ 0 Sub Move	#1500, SP FULLERROR #4, SP NEWITEM, (SP)	Check to see if the stack pointer contains an address value equals to or less than 1500. If it does, the stack is full. Branch to the routine FULLERROR for appropriate action. Otherwise, decrement the stack pointer (SP) appropriately and push the element in memory location NEWITEM onto the stack.

...

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPOP (If it allows auto increment/ decrement addressing mode)	Compare Branch > 0 Move	#2000, SP EMPTYERROR (SP)+, ITEM	Check to see if the stack pointer contains an address value greater than 2000. If it does, the stack is empty. Branch to the routine EMPTYERROR for appropriate action. Otherwise, pop the stack top element in memory location ITEM onto the stack and increment the stack pointer (SP) appropriately.

Or

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPOP (If it does not allow auto increment/ decrement addressing mode)	Compare Branch > 0 Move Add	#2000, SP EMPTYERROR (SP), ITEM #4, SP	Check to see if the stack pointer contains an address value greater than 2000. If it does, the stack is empty. Branch to the routine EMPTYERROR for appropriate action. Otherwise, pop the stack top element in memory location ITEM onto the stack and increment the stack pointer (SP) appropriately.

24. Suppose that a stack runs from location 1500 (BOTTOM) to no further than location 2000 (TOP) and each stack word consumes 4 bytes and machine is byte addressable. The stack pointer is loaded initially with the address value 1496. Write a routine for both PUSH and POP operation.
Answer:

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPUSH (If it allows auto increment/ decrement addressing mode)	Compare Branch ≥ 0 Move	#2000, SP FULLERROR NEWITEM, +(SP)	Check to see if the stack pointer contains an address value equals to or less than 1500. If it does, the stack is full. Branch to the routine FULLERROR for appropriate action. Otherwise, increment the stack pointer and push the element in memory location NEWITEM onto the stack.

Or

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPUSH (If it does not allow auto increment/ decrement addressing mode)	Compare Branch ≥ 0 Add Move	#2000, SP FULLERROR #4, SP NEWITEM, (SP)	Check to see if the stack pointer contains an address value equals to or less than 1500. If it does, the stack is full. Branch to the routine FULLERROR for appropriate action. Otherwise, increment the stack pointer (SP) appropriately and push the element in memory location NEWITEM onto the stack.

...

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPOP (If it allows auto increment/ decrement addressing mode)	Compare Branch < 0 Move	#1500, SP EMPTYERROR (SP)-, ITEM	Check to see if the stack pointer contains an address value greater than 2000. If it does, the stack is empty. Branch to the routine EMPTYERROR for appropriate action. Otherwise, pop the stack top element in memory location ITEM onto the stack and decrement the stack pointer (SP) appropriately.

Or

Name of Routine	Assemble Language		Remarks/ Actions of Assemble instructions
SAFEPOP (If it does not allow auto increment/ decrement addressing mode)	Compare Branch < 0 Move Sub	#1500, SP EMPTYERROR (SP), ITEM #4, SP	Check to see if the stack pointer contains an address value greater than 2000. If it does, the stack is empty. Branch to the routine EMPTYERROR for appropriate action. Otherwise, pop the stack top element in memory location ITEM onto the stack and decrement the stack pointer (SP) appropriately.

25. Register R5 is used in a program to point to the top of a stack. Assume that the stack address space ranges from 2000 to 1500 and each stack word consumes 4 bytes and machine is byte addressable. Write a sequence of instructions using the Index, Autoincrement, and Autodecrement addressing modes to perform each of the following tasks:

a) Pop the top two items off the stack, add them, and then push the result onto the stack.
b) Copy the fifth item from the top into register R3.
c) Remove the top ten items from the stack.

[CO/Hamacher Book/5th Ed/Exercise/2.17/Page-100]

Answer:

Here, register R5 is used as the stack pointer (SP).

a) Pop the top two items off the stack, add them, and then push the result onto the stack.

```
Move   (R5)+, R0
Add    (R5)+, R0
Move   R0, -(R5)
```

b) Copy the fifth item from the top into register R3.

```
Move 16(R5), R3
```

c) Remove the top ten items from the stack.

```
Add   #40, R5
```

Computer Organisation & Architecture Home Assignment Solution

Sem: 4TH Section: IT7, IT8 Faculty: Prof. Anil Kumar Swain

DOA: Dt.03.02.2020 DOS:Dt.14.02.2020

26. Given the following program fragment

Main Program		First Subroutine		Second Subroutine	
2000	ADD R1, R2	3000 SUB1	MOV R1,R2	4000 SUB2	SUB R6, R1
2004	XOR R3, R4	3004	ADD R5, R1	4008	XOR R1, R5
2008	CALL SUB1	3008	CALL SUB2	4012	RETURN
2012	SUB R4, R5	3012	RETURN		

Initially the stack pointer SP contains 5000.

What are the content of PC, SP, and the top of the stack?

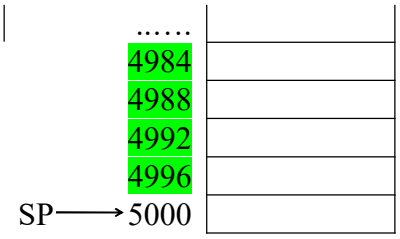
i) After the subroutine call instruction is executed in the main program?

ii) After the subroutine call instruction is executed in the subroutine SUB1?

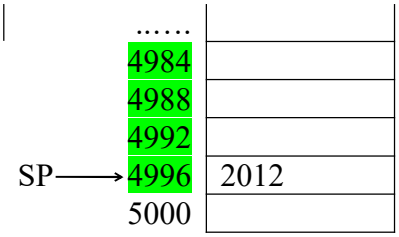
iii) After the return from SUB2 subroutine?

Answer:

Initial Condition of stack



i)After the subroutine call instruction is executed in the main program.

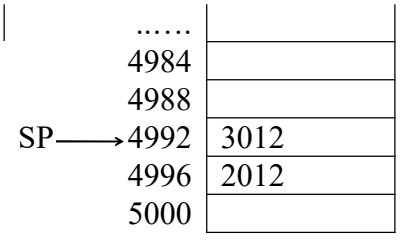


Explanation

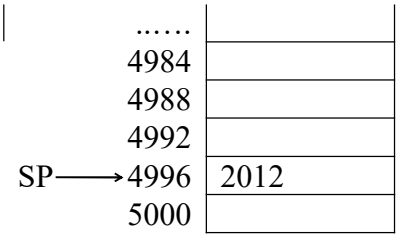
Just after the start of execution of
2008 CALL SUB1,
the address 2012 that is the address of next instruction is pushed in to the stack instead of saving it in PC, as the instruction is a sub routine call. The PC is stored as the SUB1 address. So, the content of
SP=4996
Top of Stack=[SP]=2012
PC=3000

…

ii)After the subroutine call instruction is executed in the subroutine SUB1?



iii)After the return from SUB2 subroutine?



Explanation

Explanation Just after the start of execution of 3008 CALL SUB2 , the address 3012 that is the address of next instruction is pushed in to the stack instead of saving it in PC, as the instruction is a sub routine call. The PC is stored as the SUB2 address. So, the content of SP=4992 Top of Stack=[SP]=3012 PC=4000	Just after the execution of 4012 RETURN , the stacktop address is popped and stored in PC that is PC=3012, the address of next instruction to be resumed for execution. So, the content of SP=4996 Top of Stack=[SP]=2012 PC=3012

Single Bus Organisation

27. Draw the diagram for single bus organisation of the datatpath inside a processor and explain the functionality of each of its components in one or two lines.

Answer:

Refer page no.413 of CO Book by Carl Hammacher, 5th ed.

28. Why is the Wait-for-Memory-Function-Completed (WMFC) step needed when reading from or waiting to the main memory.

[CO/Hamacher Book/5th Ed/Exercise/7.1/Page-446]

Answer:

The WMFC step is needed to synchronize the operation of the processor and the .main memory

29. Draw the diagram for the connection and control signals for register MDR.

Answer:

Refer page no.418 of CO Book by Carl Hammacher, 5th ed.

30. Write the sequence of control steps required for the single bus structure for each of the following instructions.

a) Add R3, R1
b) Add (R3), R1
c) Add 10(R3), R1

Answer:

Add R3, R1

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. R3out, Yin 5. R1out, SelectY, Add, Zin 6. Zout, R1in, End	

Add (R3), R1

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. R3out, MARin, Read 5. R1out, Yin, WMFC 6. MDRout, SelectY, Add, Zin 7. Zout, R1in, End	

Add 10(R3), R1

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. Offset field of IRout, Yin 5. R3out, Select Y, Add, Zin 6. Zout, MARin, Read 7. R1out, Yin, WMFC 8. MDRout, Select Y, Add, Zin 9. Zout, R1in, end	

31. Write the sequence of control steps required for the single bus structure for each of the following instructions.

a) Jump L1
b) Branch <0 Label1
c) Add -(R3), R1

Answer:

Jump L1
(Uncondotional Jump)

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select 4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. Offset-field-of-IRout, Add, Zin	

5. Zout, PCin, End	

Branch <0 Label1
(Condtional Jump)

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select 4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. Offset-field-of-IRout, Add, Zin if N=0, then End 5. Zout, PCin, End	

Add -(R3), R1

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select 4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. R3out, Select4, Sub, Zin 5. Zout, MARin, Read 6. R1out, Yin, WMFC 7. MDRout, Select Y, Add, Zin 8. Zout, R1in, end	

32. Write the sequence of control steps required for the single bus structure for each of the following instructions.

a) Add the (immediate) number NUM to register R1
b) Add the contents of memory location NUM to register R1
c) Add the content of memory location whose address is at memory location NUM to register R1.

Assume that each instruction consists of two words. The first word specifies the operation and the addressing mode and the second word contains the number NUM.

Answer:

(Add the (immediate) number NUM to register R1)

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. PCout, MARin, Read, Select4, Add, Zin	

5. Zout, PCin, Yin 6. R1out, Yin, WMFC 7. MDRout, SelectY, Add, Zin 8. Zout, R1in, End	

(Add the contents of memory location NUM to register R1)

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. PCout, MARin, Read, Select4, Add, Zin 5. Zout, PCin, WMFC 6. MDRout, MARin, Read 7. R1out, Yin, WMFC 8. MDRout, Add, Zin 9. Zout, R1in, End	

(Add the content of memory location whose address is at memory location NUM to register R1.)

Sequence of control steps	Explanation
1. PCout, MARin, Read, Select4, Add, Zin 2. Zout, PCin, Yin, WMFC 3. MDRout, IRin 4. PCout, MARin, Read, Select4, Add, Zin 5. Zout, PCin, WMFC 6. MDRout, MARin, Read, WMFC 7. MDRout, MARin, Read 8. R1out, Yin, WMFC 9. MDRout, Add, Zin 10. Zout, R1in, End	

33. Draw the diagram for the three bus organisation of the datatpath inside a processor and explain the functionality of each of its components in one or two lines.

Answer:

Refer page no.423 of CO Book by Carl Hammacher, 5th ed.

Computer Organisation & Architecture Home Assignment Solution

Sem: 4TH Section: IT7, IT8 Faculty: Prof. Anil Kumar Swain

DOA: Dt.03.02.2020 DOS:Dt.14.02.2020

34. Explain the working principle of Hardwired control unit design along with neat diagram. Explain its advantages and disadvantages.

Answer:

Refer page no.425 of CO Book by Carl Hammacher, 5th ed.

35. Write the control sequence for the instruction Add R1, R2, R3 for the three bus organisation.

Answer:

Add R4, R5, R6

Sequence of control steps	Explanation
1. PCout, R=B, MARin, Read, IncPC 2. WMFC 3. MDRoutB, R=B, IRin 4. R4outA, R5outB, SelectA, Add, R6in, End	

36. Draw and explain the working principle of microprogrammed control unit.

Answer:

Refer page no.429 of CO Book by Carl Hammacher, 5th ed.