USN

INTERNAL ASSESSMENT TEST – III

Sub:	Computer Organization and ARM Microcontrollers							Code:	21EC52
Date:	14/03/2024	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	ECE

Answer any 5 full questions

		Marks	CO	RBT
1.	Draw and explain the internal organization of a processor using single bus architecture. Give the control sequence for the instruction ADD (R3), R1.	[10]	CO3	L4
2	Draw and explain the multi-bus internal architecture of a processor. Give the control sequence for the instruction ADD R4, R5, R6.	[10]	CO3	L4
3	With a neat diagram, explain the organization of 2M*8 (16 Mega-bits) dynamic memory chip	[10]	CO2	L2
4	Write short notes on: a) Hit rate and Miss Penalty associated with CACHE b) Virtual Memory c) Locality of Reference d) Replacement Algorithms for CACHE.	[10]	CO2	L2

USN

INTERNAL ASSESSMENT TEST – III

Sub:	Computer Organization and ARM Microcontrollers							Code:	21EC52
Date:	14/03/2024	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	ECE

Answer any 5 full questions

		Marks	CO	RBT
1.	Draw and explain the internal organization of a processor using single bus architecture. Give the control sequence for the instruction ADD (R3), R1.	[10]	CO3	L4
2	Draw and explain the multi-bus internal architecture of a processor. Give the control sequence for the instruction ADD R4, R5, R6.	[10]	CO3	L4
3	With a neat diagram, explain the organization of 2M*8 (16 Mega-bits) dynamic memory chip	[10]	CO2	L2
4	Write short notes on: a) Hit rate and Miss Penalty associated with CACHE b) Virtual Memory c) Locality of Reference d) Replacement Algorithms for CACHE.	[10]	CO2	L2

		Marks	CO	RBT
5	With a neat diagram, explain the working of a microprogrammed control unit	[10]	CO3	L2
6	Differentiate Hardwired Control against Microprogrammed Control	[10]	CO3	L4
7	a) List and detail the various basic C data types used. b) Write an ALP to find the count of negative numbers from an array of 5 numbers (Assume suitable data).	[05] [05]	CO4	L3
8	List the various types of Mapping Functions associated with CACHE memory. Explain any one in detail.	[10]	CO2	L2

CI CCI HOD

		Marks	CO	RBT
5	With a neat diagram, explain the working of a microprogrammed control unit	[10]	CO3	L2
6	Differentiate Hardwired Control against Microprogrammed Control	[10]	CO3	L4
7	a) List and detail the various basic C data types used. b) Write an ALP to find the count of negative numbers from an array of 5 numbers (Assume suitable data).	[05] [05]	CO4	L3
8	List the various types of Mapping Functions associated with CACHE memory. Explain any one in detail.	[10]	CO2	L2

CI CCI HOD

SOLUTION:

ANS-1]

SINGLE BUS ORGANIZATION

- ALU and all the registers are interconnected via a **Single Common Bus** (Figure 7.1).
- Data & address lines of the external memory-bus is connected to the internal processor-bus via MDR& MAR respectively. (MDR→ Memory Data Register, MAR → Memory Address Register).
- **MDR** has 2 inputs and 2 outputs. Data may be loaded
  → into MDR either from memory-bus (external) or
  → from processor-bus (internal).
- **MAR"**s input is connected to internal-bus; MAR"s
  output is connected to external-bus.
- **Instruction Decoder & Control Unit** is responsible for
  → issuing the control-signals to all the units inside the processor.
  → implementing the actions specified by the instruction (loaded in the IR).
- Register R0 through R(n-1) are the **Processor Registers**.
  The programmer can access these registers for general-purpose use.
- Only processor can access 3 registers **Y**, **Z** & **Temp** for temporary storage during program-execution.The programmer cannot access these 3 registers.
- In **ALU**, 1) „A" input gets the operand from the output of the multiplexer (MUX).
  2) „B" input gets the operand directly from the processor-bus.
- There are 2 options provided for „A" input of the ALU.
- MUX is used to select one of the 2 inputs.
- **MUX** selects either
  → output of Y or
  → constant-value 4(which is used to increment PC content).

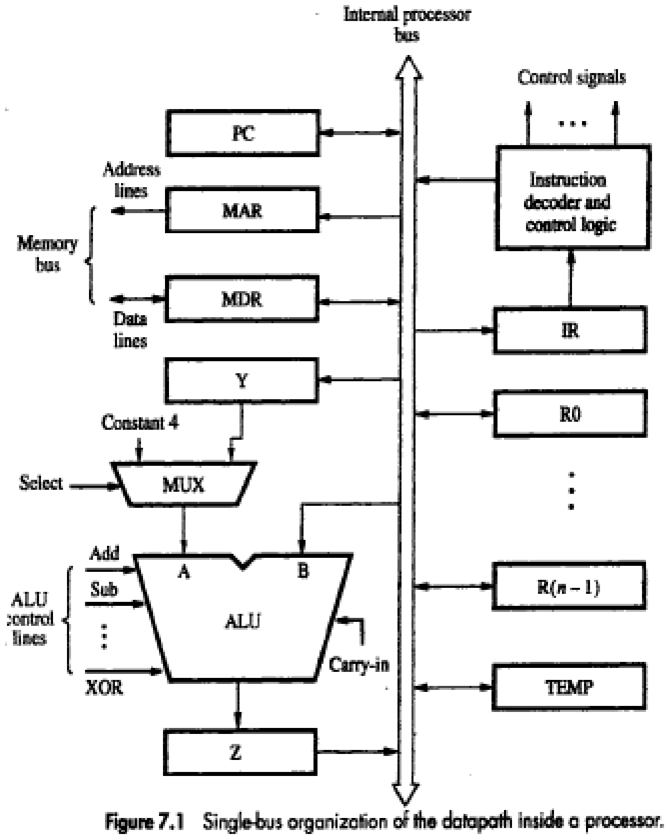


Figure 7.1 Single-bus organization of the datapath inside a processor.

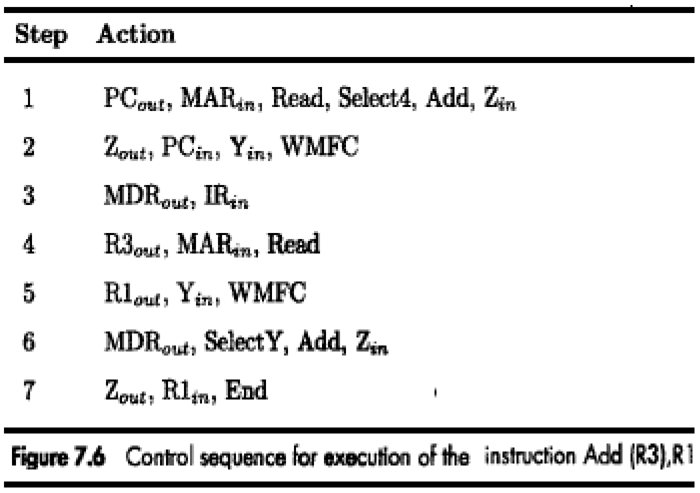
- An instruction is executed by performing one or more of the following operations:
  1) Transfer a word of data from one register to another or to the ALU.
  2) Perform arithmetic or a logic operation and store the result in a register.
  3) Fetch the contents of a given memory-location and load them into a register.

4) Store a word of data from a register into a given memory-location.

- **Disadvantage:** Only one data-word can be transferred over the bus in a clock cycle.

Solution: Provide multiple internal-paths. Multiple paths allow several data-transfers to take place in parallel

CONTROL SEQUENCE FOR ADD (R3), R1:



Step	Action
1	PC_{out}, MAR_{in}, Read, Select4, Add, Z_{in}
2	Z_{out}, PC_{in}, Y_{in}, WMFC
3	MDR_{out}, IR_{in}
4	$R3_{out}$, MAR_{in}, Read
5	$R1_{out}$, Y_{in}, WMFC
6	MDR_{out}, SelectY, Add, Z_{in}
7	Z_{out}, $R1_{in}$, End

Figure 7.6 Control sequence for execution of the instruction Add (R3),R1

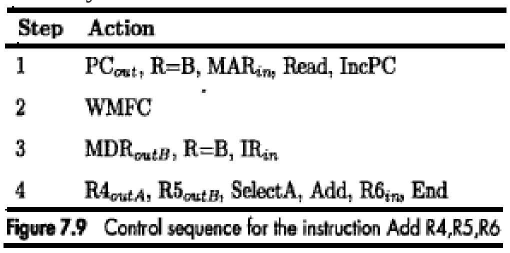
ANS-2:

MULTIPLE BUS ORGANIZATION

- **Disadvantage of Single-bus organization:** Only one data-word can be transferred over the bus ina clock cycle. This increases the steps required to complete the execution of the instruction

Solution: To reduce the number of steps, most processors provide multiple internal-paths. Multiplepaths enable several transfers to take place in parallel.

- As shown in fig 7.8, three buses can be used to connect registers and the ALU of the processor.
- All general-purpose registers are grouped into a single block called the **Register File**.
- Register-file has 3 ports:
  1) Two output-ports allow the contents of 2 different registers to be simultaneously placed onbuses A & B.
  2) Third input-port allows data on bus C to be loaded into a third register during the sameclock-cycle.
- Buses A and B are used to transfer source-operands to A & B inputs of ALU.
- The result is transferred to destination over bus C.
- **Incrementer Unit** is used to increment PC by 4.



Step	Action
1	PC_{out}, R=B, MAR_{in}, Read, IncPC
2	WMFC
3	MDR_{outB}, R=B, IR_{in}
4	$R4_{outA}$, $R5_{outB}$, SelectA, Add, $R6_{in}$, End

Figure 7.9 Control sequence for the instruction Add R4,R5,R6

- Instruction execution proceeds as follows:

Step 1--> Contents of PC are
→ passed through ALU using R=B control-signal &
→ loaded into MAR to start memory Read operation. At the same time, PC is incremented by 4.

Step2--> Processor waits for MFC signal from memory.

Step3--> Processor loads requested-data into MDR, and then transfers them to IR.Step4--> The instruction is

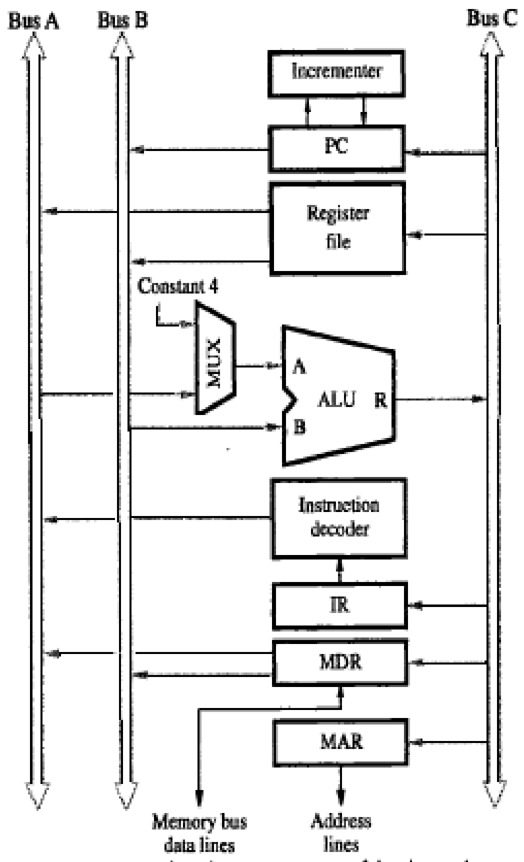


Figure 7.8 Three-bus organization of the datapath.

decoded and add operation takes place in a single Step.

ANS-3:

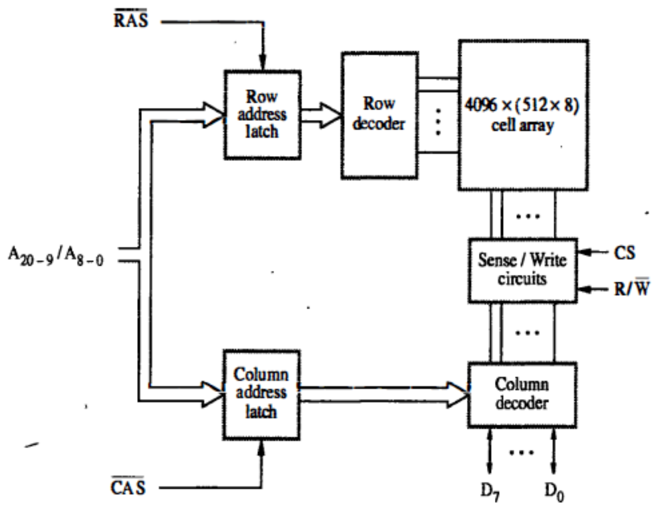


Figure 5.7 Internal organization of a 2M × 8 dynamic memory chip.

- The high-order 12 bits constitute the row address and the low-order 9 bits of the address constitute column address of a byte.

●To reduce the number of pins needed for external connections, the row and column addresses are multiplexed on 12 pins.

●During a Read or a Write operation, the row address is applied first.

●It is loaded into the row address latch in response to a signal pulse on the Row Address Strobe (RAS) input of the chip.

●Then a Read operation is initiated, in which all cells on the selected row are read and refreshed.

●Shortly after the row address is loaded, the column address is applied to the address pins and loaded into the column address latch under control of the Column Address Strobe (CAS) signal.

●The information in this latch is decoded and the appropriate group of 8 Sense/Write circuits are selected.

●If the R/LΩWcontrol signal indicates a Read operation, the output values of the selected circuits are transferred to the data lines, D7-0.

●For a Write operation, the information on the D7-0lines is transferred to the selected circuits.
●This information is then used to overwrite the contents of the selected cells in the corresponding 8 columns.

In commercial DRAM chips, the RAS and CAS control signals are active low.
●To indicate this fact, these signals are shown on diagrams as RASand CAS.

●To ensure that the contents of a DRAM are maintained, each row of cells must be accessed periodically.
●A *refresh circuit* usually performs this function automatically.

ANS-4:

HIT AND MISS PENALTY ASSOCIATED WITH CACHE:

A successful access to data in a cache is called a hit. The number of hits stated as a fraction of all attempted accesses is called the *hit rate*, and the *miss rate* is the number of misses stated as a fraction of attempted accesses.

Ideally, the entire memory hierarchy would appear to the processor as a single memory unit that has the access time of the cache on the processor chip and the size of the magnetic disk. How close we get to this ideal depends largely on the hit rate at different levels of the hierarchy.

High hit rates well over 0.9 are essential for high-performance computers. Performance is adversely affected by the actions that need to be taken when a miss occurs.

A performance penalty is incurred because of the extra time needed to bring a block of data from a slower unit in the memory hierarchy to a faster unit. During that period, the processor is stalled waiting for instructions or data. The waiting time depends on the details of the operation of the cache.

For example, it depends on whether or not the load-through approach is used. We refer to the total access time seen by the processor when a miss occurs as the *miss penalty*.

Consider a system with only one level of cache. In this case, the miss penalty consists almost entirely of the time to access a block of data in the main memory. Let h be the hit rate, M the miss penalty, and C the time to access information in the cache. Thus, the average access time experienced by the processor is

$t_{avg} = hC + (1 - h)M$

The following example illustrates how the values of these parameters affect the average access time.

Consider a computer that has the following parameters. Access times to the cache and the main memory are τ and 10τ, respectively. When a cache miss occurs, a block of 8 words is transferred from the main memory to the cache. It takes 10τ to transfer the first word of the block, and the remaining 7 words are transferred at the rate of one word every τ seconds.

The miss penalty also includes a delay of τ for the initial access to the cache, which misses, and another delay of τ to transfer the word to the processor after the block is loaded into the cache (assuming no load-through).

Thus, the miss penalty in this computer is given by:

$M = \tau + 10\tau + 7\tau + \tau = 19\tau$

Virtual memory:

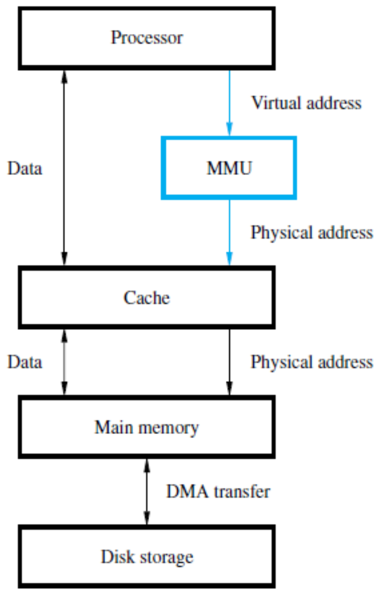
An important challenge in the design of a computer system is to :

- provide a large, fast memory system at an affordable cost.
- Architectural solutions to increase the effective speed and size of the memory system.
- Cache memories were developed to increase the effective speed of the memory system.
- Virtual memory is an architectural solution to increase the effective size of the memory system.

Basic Requirement of Virtual Memory & Its Use:

- In most modern computer systems, the physical main memory is not as large as the address space spanned by an address issued by the processor.
- For example, a processor that issues 32-bit addresses has an addressable space of 4G bytes.
- The size of the main memory in a typical computer ranges from a few hundred megabytes to 1G bytes.
- When a program does not completely fit into the main memory, the parts of it not currently being executed are stored on secondary storage devices, such as magnetic disks.
- All parts of a program that are eventually executed are first brought into the main memory.
  - When a new segment of a program is to be moved into a full memory, it must replace another segment already in the memory.
- In modern computers, the operating system moves programs and data automatically between the main memory and secondary storage.
- Thus, the application programmer does not need to be aware of limitations imposed by the available main memory.
- Techniques that automatically move program and data blocks into the physical main memory when they are required for execution are called *virtual-memory* techniques.
- Programs, and hence the processor, reference an instruction and data space that is independent of the available physical main memory space.
- The binary addresses that the processor issues for either instructions or data are called *virtual* or *logical addresses*.
- These addresses are translated into physical addresses by a combination of hardware and software components.
- If a virtual address refers to a part of the program or data space that is currently in the physical memory, then the contents of the appropriate location in the main memory are accessed immediately.

- If the referenced address is not in the main memory, its contents must be brought into a suitable location in the memory before they can be used.



Virtual Memory Organization

- A special hardware unit, called the *Memory Management Unit*(MMU), translates virtual addresses into physical addresses.
- When the desired data (or instructions) are in the main memory, these data are fetched.
- If the data are not in the main memory, the MMU causes the operating system to bring the data into the memory from the disk.
- Transfer of data between the disk and the main memory is performed using the DMA scheme.

LOCALITY OF REFERENCE:

The cache is a small and very fast memory, interposed between the processor and the main memory. Its purpose is to make the main memory appear to the processor to be much faster than it actually is.

The effectiveness of this approach is based on a property of computer programs called ***locality of reference*.**

Analysis of programs shows that most of their execution time is spent in routines in which many instructions are executed repeatedly. These instructions may constitute a simple loop, nested loops, or a few procedures that repeatedly call each other. The actual detailed pattern of instruction sequencing is not important—the point is that many instructions in localized areas of the program are executed repeatedly during some time period.

This behavior manifests itself in two ways: **temporal and spatial.**

- The first means that a recently executed instruction is likely to be executed again very soon.
- The spatial aspect means that instructions close to a recently executed instruction are also likely to be executed soon.

Conceptually, operation of a cache memory is very simple. The memory control circuitry is designed to take advantage of the property of locality of reference.

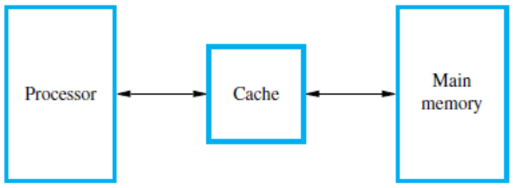
Temporal locality suggests that whenever an information item, instruction or data, is first needed, this item should be brought into the cache, because it is likely to be needed again soon.

Spatial locality suggests that instead of fetching just one item from the main memory to the cache, it is useful to fetch several items that are located at adjacent addresses as well.

REPLACEMENT ALGORITHMS FOR CACHE:

The term *cache block* refers to a set of contiguous address locations of some size. Another term that is often used to refer to a cache block is a *cache line*.

Consider the arrangement in Figure 8.15.



When the processor issues a Read request, the contents of a block of memory words containing the location specified are transferred into the cache.

Subsequently, when the program references any of the locations in this block, the desired contents are read directly from the cache. Usually, the cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory.

The correspondence between the main memory blocks and those in the cache is specified by a *mapping function*. When the cache is full and a memory word (instruction or data) that is not in the cache is referenced, the cache control hardware must decide which block should be removed to create space for the new block that contains the referenced word.

The collection of rules for making this decision constitutes the cache's *replacement algorithm*.

When a new block is to be brought into the cache and all the positions that it may occupy are full, the cache controller must decide which of the old blocks to overwrite.

This is an important issue, because the decision can be a strong determining factor in system performance. In general, the objective is to keep blocks in the cache that are likely to be referenced in the near future. But, it is not easy to determine which blocks are about to be referenced. The property of locality of reference in programs gives a clue to a reasonable strategy.

Because program execution usually stays in localized areas for reasonable periods of time, there is a high probability that the blocks that have been referenced recently will be referenced again soon. Therefore, when a block is to be overwritten, it is sensible to overwrite the one that has gone the longest time without being referenced.
This block is called the *least recently used* (LRU) block, and the technique is called the *LRU replacement algorithm*.

The LRU algorithm has been used extensively. Although it performs well for many access patterns, it can lead to poor performance in some cases. For example, it produces disappointing results when accesses are made to sequential elements of an array that is slightly too large to fit into the cache.

Performance of the LRU algorithm can be improved by introducing a small amount of randomness in deciding which block to replace.

Several other replacement algorithms are also used in practice. An intuitively reasonable rule would be to remove the "oldest" block from a full set when a new block must be brought in. However, because this algorithm does not take into account the recent pattern of access to blocks in the cache, it is generally not as effective as the LRU algorithm in choosing the best blocks to remove.

The simplest algorithm is to randomly choose the block to be overwritten. Interestingly enough, this simple algorithm has been found to be quite effective in practice.

ANS-5: Working of a microprogrammed control unit:

MICROPROGRAMMED CONTROL

- Microprogramming is a method of control unit design (Figure 7.16).
- Control-signals are generated by a program similar to machine language programs.
- **Control Word(CW*)*** is a word whose individual bits represent various control-signals (like Add, PC_{in}).
- Each of the control-steps in control sequence of an instruction defines a unique combination of 1s &0s in CW.
- Individual control-words in microroutine are referred to as **microinstructions** (Figure 7.15).

• A sequence of CWs corresponding to control-sequence of a machine instruction constitutes the **microroutine.**

• The microroutines for all instructions in the instruction-set of a computer are stored in a specialmemory called the **Control Store (CS)**.

• Control-unit generates control-signals for any instruction by sequentially reading CWs ofcorresponding microroutine from CS.

• **µPC** is used to read CWs sequentially from CS. (µPC→ Microprogram Counter).

• Every time new instruction is loaded into IR, o/p of **Starting Address Generator** is loaded into µPC.

• Then, µPC is automatically incremented by clock;

causing successive microinstructions to be read from CS.

Hence, control-signals are delivered to various parts of processor in correct sequence.

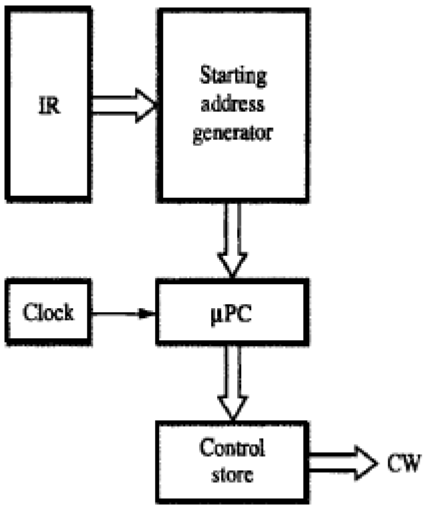
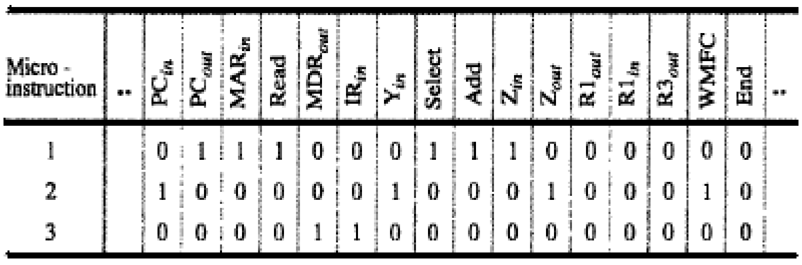


Figure 7.16 Basic organization of a microprogrammed control unit.



Micro - instruction	..	PC_{in}	PC_{out}	MAR_{in}	Read	MDR_{out}	IR_{in}	Y_{in}	Select	Add	Z_{in}	Z_{out}	$R1_{out}$	$R1_{in}$	$R3_{out}$	WMFC	End	..
1		0	1	1	1	0	0	0	1	1	1	0	0	0	0	0	0	
2		1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	
3		0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	

Figure 7.15 An example of microinstructions for Figure 7.6.

Advantages

• It simplifies the design of control unit. Thus it is both, cheaper and less error prone implement.

• Control functions are implemented in software rather than hardware.

• The design process is orderly and systematic.

• More flexible, can be changed to accommodate new system specifications or to correct the designerrors quickly and cheaply.

• Complex function such as floating point arithmetic can be realized efficiently.

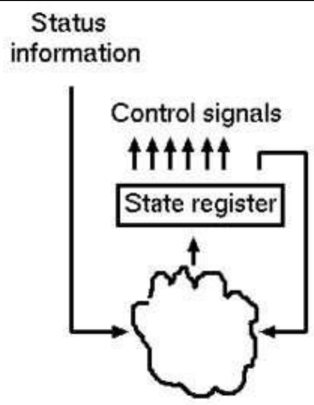
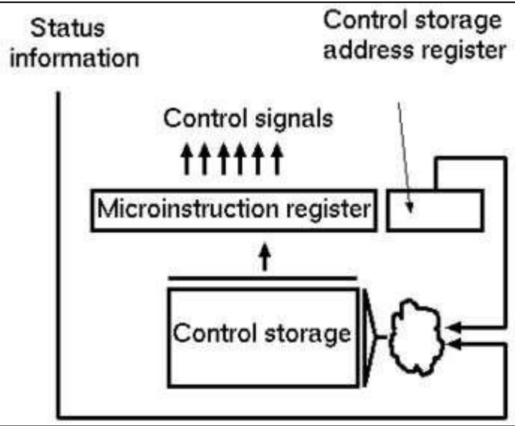
Disadvantages

• A microprogrammed control unit is somewhat slower than the hardwired control unit, because time is required to access the microinstructions from CM.

The flexibility is achieved at some extra hardware cost due to the control memory and its accesscircuitry.

ANS-6: Differentiate Hardwired Control against Microprogrammed Control

Attribute	**Hardwired Control**	**Microprogrammed Control**
Definition	Hardwired control is a control mechanism to generate control-signals by using gates, flip-flops, decoders, and other digital circuits.	Micro programmed control is a control mechanism to generate control-signals by using a memory called control store (CS), which contains the control-signals.

Speed	Fast	Slow
Control functions	Implemented in hardware.	Implemented in software.
Flexibility	Not flexible to accommodate new system specifications or new instructions.	More flexible, to accommodate new system specification or new instructions redesign is required.
Ability to handle large or complex instruction sets	Difficult.	Easier.
Ability to support operating systems & diagnostic features	Very difficult.	Easy.
Design process	Complicated.	Orderly and systematic.
Applications	Mostly RISC microprocessors.	Mainframes, some microprocessors.
Instructionset size	Usually under 100 instructions.	Usually over 100 instructions.
ROM size	-	2K to 10K by 20-400 bit microinstructions.
Chip area efficiency	Uses least area.	Uses more area.
Diagram		 Status information Control storage address register Control signals Microinstruction register Control storage

ANS-7: List and detail the various basic C data types used.

- Numeric Data Types (int, short, long, long long, float, double, long double)
- Character Data Types (Char)
- Unsigned Data Types (Unsigned int, Unsigned short, Unsigned Long, Unsigned long long,
- Boolean Data Type (_Bool)
- Complex Data Type (_Complex)
- Imaginary Data Type (_Imaginary)

Four Basic Data Types are:

- Int
- Float
- Double
- Char

ANS-8:

Various types of Mapping Functions associated with CACHE memory. Explain any one in detail.

Mapping Functions associated with CACHE:

- Direct-mapped Cache
- Associative-mapped Cache
- Set-associative-mapped Cache

Explain Any One Technique:

1) Direct-mapped Cache

Consider a cache consisting of 128 blocks of 16 words each, for a total of 2048 (2K) words, and assume that the main memory is addressable by a 16-bit address. The main memory has 64K words, which we will view as 4K blocks of 16 words each. For simplicity, we have assumed that consecutive addresses refer to consecutive words.

Direct Mapping

The simplest way to determine cache locations in which to store memory blocks is the *direct-mapping* technique. In this technique, block *j* of the main memory maps onto block *j* modulo 128 of the cache, as depicted in Figure 8.16.

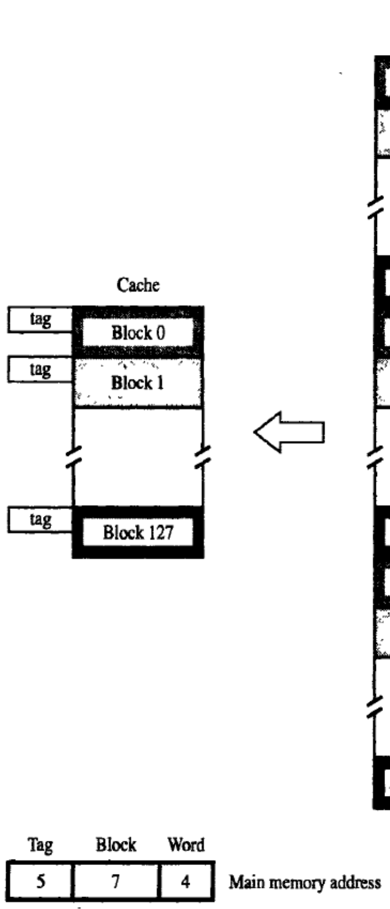


Fig. 8.16

Thus, whenever one of the main memory blocks 0, 128, 256, . . . is loaded into the cache, it is stored in cache block 0. Blocks 1, 129, 257, . . . are stored in cache block 1, and so on. Since more than one memory block is mapped onto a given cache block position, contention may arise for that position even when the cache is not full. For example, instructions of a program may start in block 1 and continue in block 129, possibly after a branch. As this program is executed, both of these blocks must be transferred to the block-1 position in the cache. Contention is resolved by allowing the new block to overwrite the currently resident block. With direct mapping, the replacement algorithm is trivial. Placement of a block in the cache is determined by its memory address. The memory address can be divided into three fields, as shown in Figure 8.16.

The low-order 4 bits select one of 16 words in a block. When a new block enters the cache, the 7-bit cache block field determines the cache position in which this block must be stored.
The high-order 5 bits of the memory address of the block are stored in 5 *tag* bits associated with its location in the cache. The tag bits identify which of the 32 main memory blocks mapped into this cache position is currently resident in the cache.

As execution proceeds, the 7-bit cache block field of each address generated by the processor points to a particular block location in the cache. The high-order 5 bits of the address are compared with the tag bits associated with that cache location. If they match, then the desired word is in that block of the cache.

If there is no match, then the block containing the required word must first be read from the main memory and loaded into the cache. The direct-mapping technique is easy to implement, but it is not very flexible.