Computer Architecture : Assignment 02
23K-2001

Answer#01:

(a)

Data Registers: They are general purpose registers mainly used to store temporary data during execution. In x86 architectures, the registers are as follows:

AX : Accumulator register → used for arithmetic operations.

BX : Base register → used for indexed addressing

CX : Counter register → used in loops/shift operations.

DX : Data register → used in input/output operations & * or /.

Pointer Registers: They are used to store memory addresses and control memory access. The registers are as follows:

SP : Stack Pointer → points to the top of the stack.

BP : Base Pointer → used to access function parameters & local variables in stack frames.

SI : Source Index → used in string & memory operations.

DI : Destination Index → used in string & memory operations along SI.

IP : Instruction Pointer → stores offset of next instruction to be executed.

extended (32-bit)	16-bit	8-bit
EAX	AX	AH, AL
EBX	BX	BH, BL
ECX	CX	CH, CL
EDX	DX	DH, DL
ESP	SP	-
EBP	BP	-
ESI	SI	-
EDI	DI	-
EIP	IP	-

(b)

Purpose of Segment Registers

Code Segment Register: stores starting address of code segment & tells CPU where current program instructions are stored.

Data Segment Register: stores starting address of data segment & used to access global and static variables.

Extra Segment Register: additional data segment register, used for string operations.

Stack Segment Register: stores starting address of the stack, used in managing function calls, returns and local variables.

(c)

Memory Management Unit:

The MMU is a core hardware component in a computer architecture, used by the computer programs. The function of MMU are as follows:

→ Translates virtual addresses into physical addresses.

→ Controls memory protection and access rights.

→ Helps isolate processes in multitasking environments.

→ Enables virtual memory implementation such as paging /segmentation.

→ Boosts security and stability by preventing unauthorized access.

Answer #02:

(a)

Key Characteristics of RISC Architecture:

i. Simple Instructions: RISC uses fewer and simpler instructions that execute in one cycle.

ii. Load/Store Architecture: Memory access is limited to specific load and store instructions.

iii. Large number of registers: Allows storage of more variables in registers, reducing memory access.

iv. Fixed-length Instructions: Simplifies instruction decoding and pipeline design.

v. Efficient Pipelining: Due to uniform instruction size & execution time.

vi. Simple Addressing: Most of the instructions use simple register addressing.

(b)

Performance benefits of RISC Architecture:

i. Faster execution: simpler instructions reduce execution time, allowing to be executed in a single clock cycle.

ii. Efficient pipeling: fixed instruction size and uniform instruction format allows efficient pipelining, processing multiple instructions simultaneously.

iii. Simpler hardware design: small instruction set in RISC simplifies the control logic, reducing the complexity and cost of hardware implementation.

iv. Higher instruction throughput: RISC systems can execute more instructions per second due to consistent and rapid execution.

v. Compiler optimization: regular instruction format makes it easier for compilers to generate efficient machine code.

vi. Low power consumption: simpler instructions and streamlined hardware contribute to reduced power usage.

Answer#03:

a.

Data Transfer Operations: used in moving or copying of data. The data is not altered i.e no arithmetic or logical operations are performed. The data is moved between registers, memory and I/O ports.
Examples: MOV, PUSH, POP, LOAD, STORE.

b.

Logical Operations: used to perform bitwise logic operations on binary data. The data bits are manipulated to perform comparisons, masking, conditional operations and setting/toggling.
Examples: AND, OR, XOR, NOT

Answer#04:

a.

Data Types of x86 Architecture:

- Byte (8 bit): smallest data unit, used for char
- Word (16-bit): two bytes, the default size for many 8086 instructions.
- Double word (32-bit): 4 bytes, used in modern x86 processors.
- Quad word (64-bit): 8 bytes, used in 64 bit systems for high-precision calculations.
- BCD & Floating Point: Binary coded decimal and floating point, used to represent real numbers with single-precision (32 bit) and double precision (64-bit).

b.

Processor Actions on Various operations:

- Data transfer: move data between registers, memory or I/O.
- Arithmetic operation: perform add, subtract, multiply or divide.
- Logical operation: perform bitwise operations such as AND, OR, XOR, NOT.
- I/O operations: perform read/write to I/O ports.
- Control operations: alter the flow of execution (eg: jump, calls).
- Shift/Rotate Operations: shift or rotate bits within a register.

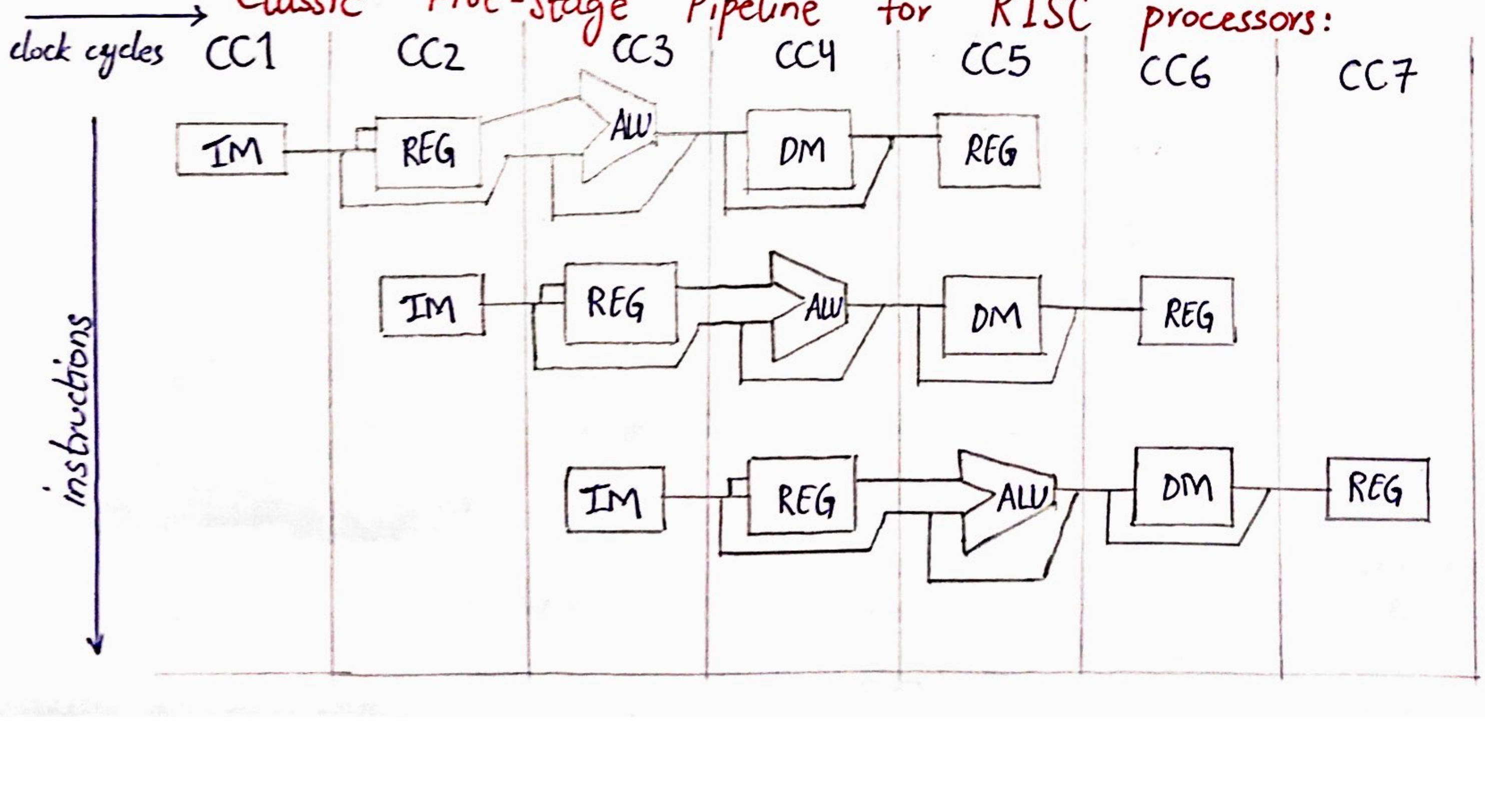
Answer#05:

(a)

RISC processor	CISC processor
• Simple and few instructions.	• Complex instructions.
• One clock cycle to execute instructions.	• Varying number of clock cycles.
• Memory access is limited to load/store instructions.	• Multiple memory access per instruction.
• Pipelining is easier to implement.	• Pipelining is difficult to implement.
• Fixed length instructions.	• Variable length instructions.
• Hardware required is simpler and easy to implement.	• Hardware required is complex and larger.
• eg: ARM.	• eg: x86

(b)

Classic Five-Stage Pipeline for RISC processors:



(c)

Data Hazards:

RAW	WAR	WAW
• Read After Write.	• Write After Read	• Write After Write
• Known as true dependency.	• known as anti-dependency.	• known as output dependency.
• instruction depends on the result of previous instruction.	• instructions writes to register before the previous instruction has read it.	• two instructions write to the same register.
eg: ADD R1, R2, R3 SUB R4, R1, R5 ← RAW hazard	eg: SUB R4, R1, R5 ADD R1, R2, R3 ← WAR hazard	eg: MOV R1, R2 ADD R1, R3, R4 ← WAW hazard

23K-2001