Assignment 3

Q1) 15 Generations

Nehalem, Sandy Bridge, Ivy Bridge, Haswell, Broadwell, Skylake, Kaby Lake, Coffee Lake, Amber Lake, Whiskey Lake, Cannon Lake, Cascade Lake, Ice Lake, Comet Lake, Tiger Lake.

Q2) It has a seperate 8087 Floating Point Math Unit.

Q3)

Processor	4004	8080	8086	80386	80686
Data Bus	4 bit	8 bit	8 bit	32 bit	64 bit
Address Bus	12 bit	14 bit	16 bit	32 bit	64 bit
Cache	N/A	N/A	N/A	N/A	16-32 KiB
Clock Rate	400-800 KHz	500-800 KHz	2 MHz	16-33 MHz	3.8+ GHz

Q4) a) Intel 8008

b) Intel 80286

c) 80486

d) 80286

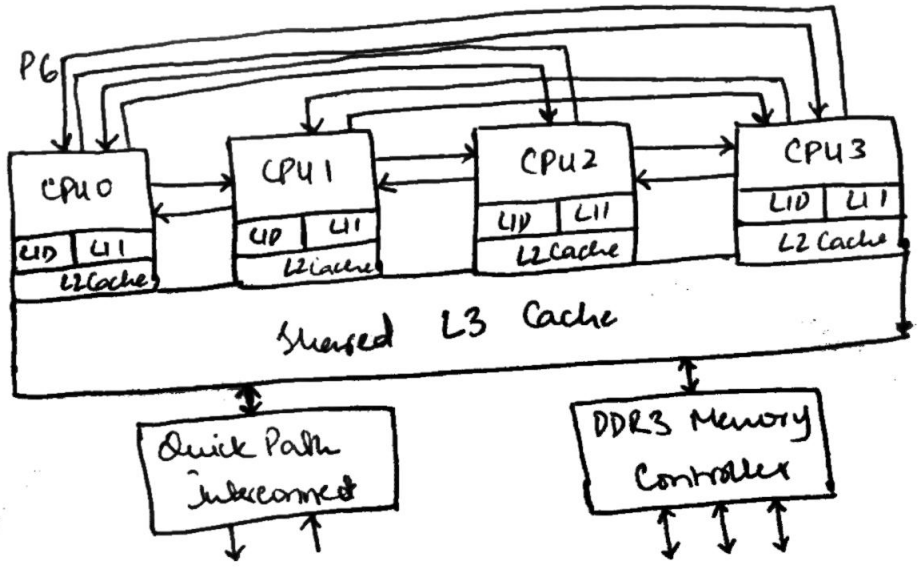
e) 80586

f) 80686

g) 80686

Q5) Intel P6

Q6)



Q7) They are almost the same.
Architecture is the same but one core is disabled in Core Solo so it can be sold at a lower price.

Q8)

Processor	i3	i5	i7
Core Count	2	4	8
Turbo Boost	No	Yes	Yes
Hyper Threading	Yes	No	Yes

Q9) Intel® Core™ i7 - 1065 U

- Intel® Core™ — Brand
- i7 — Brand Modifier
- 1065 — Generation Indicator
- 6 — SKU
- U — Product Line Suffix

Q10) $\frac{\text{time}}{\text{program}} = \frac{10,000,000}{1} \times \left(\frac{0.4 \times 10,000,000}{3} + \frac{0.35 \times 10,000,000}{5} + \frac{0.25 \times 10,000,000}{2} \right) \times \frac{1}{3 \times 10^9}$

$=$

Q11) Single Cycle CPUs execute each instruction in one cycle.
Easy to implement.
Has to operate at the speed of the slowest instruction.

Multi Cycle CPUs execute instructions in multiple cycles.
Harder to implement
Can execute instructions at varying rate.

Q12) Easier to implement (Advantage)
Must operate at the speed of the slowest instruction (Disadvantage)

Q13) Single Cycle CPUs are inefficient

Q14)

Q15) In even pipelines all instructions proceed in sync whereas in uneven pipelines some instructions might complete first and proceed to the next step while some still might not have completed.

Q16) Store the result of the previously executed instruction.

Q17) Structural Hazard= two or more instruction request the same resource.
Data Hazard = Running instruction(s) need result of another incomplete instruction.
Control Hazard= Occur when instructions change the PC.

Q18) Data Hazard

Q19)

Q20)

RISC	CISC
Fixed Length Instructions	
All instructions take same number of cycles to complete.	← Opposite of that
Simpler instructions	

Section 2

Q1) Segmented Memory Model divides the memory into segments reffered to by segment registers which store pointers. The 8086 used segmented model so it could utilize a larger amount of memory than its 16 bit registers allowed.

Data and Stack related operations can be performed seperately from code related operations to improve security.

Different processes can share data if they need to.

Q2) Real Mode

Only allows access to 1MiB of memory

CS:IP addressing mode is used.

Protected Mode

Allows memory to be addressed by 32 bits.

Access to segment registers is limited to the OS.

Supports larger data structures than Real Mode.

Long Mode

All registers are of 64 bits wide.

48 bits can be used to address memory.

Q4)

Processor	Data Bus	Address Bus
8080	8 bit	14 bit
8086	8 bit	16 bit
80386	32 bit	32bit
x86-64	64 bit	64 bit

Q5) It has 16 bit registers which can only address 64 KiB of memory.

Q6) 0xA5B8 * 0x10 + 0xD4E6 = 0xA5B80 + 0xD4E6 = 0x B3066

Q7) Like this ↗

Q8) Sign (S) = Set if calculation result is negative

Zero(Z) = Set if result is zero

Parity(P) = Set if number of 1 bits is even.

Carry(C) = Set if last operation generated a carry or borrow.

Auxiliary(AC) = Used for BCD calculations.

Q9) CS = Points to the start of the Code Segment

DS = " " " " " " Data Segment

SS = " " " " " " Stack Segment

ES = " " " " " " Extra Segment

BP = Points to the bottom of the stack

SP = Points to the top of the stack

SI = Points to a source inside the Data Segment

DI = Points to a destination inside the Extra Segment

IP = Points to an instruction in the Code Segment

Q10) 16 exabytes of RAM can be addressed by 64 bits. Since this amount of RAM isn't manufactured and we won't need all 64 bits to address current amounts of memory only 48 bits are made addressable. Making all 64 bits addressable will make pointer calculation more complex and use more energy.

Q11)

Advantages

Fast, very

Memory is utilized very efficiently

Takes less amount of storage space

Disadvantages

Hard to write

Hard to understand

Hard to debug

Not portable / crossplatform

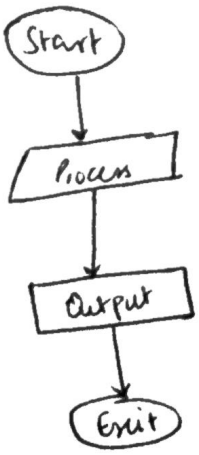
Q12) NASM is an assembler and disassembler for intel x86 architecture. It can produce multiple types of binary formats like ELF and COFF NASM does not produce executables (other than Com). NASM output needs to be linked to executables.

Q13) x86-64 is a CISC architecture and some instructions have multiple operands while some dont.

Q14) Linker = Links object files into one executable program.

Loader = loads programs and libraries into memory and prepares them for execution.

Q15)



Section 3

Q1) Assembler Instructions = These instructions are converted into machine code.

Pseudo Instructions = Assembler specific instructions.

Assembler Directives = Direct the assembler to do something

Q2) Moves data from one storage space to another.

mov register, immediate

mov rax, 0x4141414141414141

mov register, register

mov rax, rsp

mov memory, register

mov qword [var], rax

mov register, memory

mov rax, qword [var]

Q3) register = rdi

immediate = 0xdeadbeef

memory = dword [var]

Q4) To specify the entry point of the program

Q5) .data = stores initialized constants and variables

.bss = stores uninitialized data.

.text = stores code.

Q6) A library function which requests a service from the kernel.

Q7)
```
SECTION .bss
  mystring resb 17
```

Q8)
```
SECTION .data
EXIT_STATUS equ 54

SECTION .bss
 buf resb 17

SECTION .text
 global _start
 _start
  mov rax, 0
  mov rdi, 0
  mov rsi, buf
  mov rdx, 17
  syscall
  mov rax,
  mov rdi, EXIT_STATUS
  syscall
```

Q9)
```
SECTION .data
EXIT_STATUS equ 49
```

Q10) 0

Q11) Set breakpoints
Pause execution
Check state of variables
Examine the stack
Examine the registers.

Q12) To debug their code.

Q13) shell <command>

Q14) info <scope> copyString

Q15) L 1, 16

Q16) set variable var1 = 100

Q17) set disassembly-flavour intel

Q18) $ A variable used for debugging purposes.

set $ex = $esp
print $ex

Q19) break <line_number>
break 8

break <function>
break -start

break <address>
break 0xdeadbeef

break on condition
break 10 if i == 4

Q20) p <function>
call <function>
jump <function>