**Question #1 (15 points):**

Execute the program above at PC = 200 and show the detailed changes of the Control Bits in the table below. (only use zeros and ones)

| Clock Cycle | C17 | C16 | C15 | C14 | C13 | C12 | C11 | C10 | C9 | C8 | C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Cycle 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| Cycle 2 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| Cycle 3 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| Cycle 4 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| Cycle 5 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

**Question #2 (10 points):**

Execute the program above at PC = 200 and show the detailed changes of the Control Registers in the table below. (Use only characters and numbers not binary)

| | PC | IR | MAR | MBR |
|---|---|---|---|---|
| Cycle 1 | 206 | AND $R_2, R_1$ | 200 | AND $R_2, R_1$ |
| Cycle 2 | 208 | ADD $R_4, R_6$ | 204 | ADD $R_4, R_6$ |
| Cycle 3 | 212 | SUB $R_3, R_0$ | 208 | SUB $R_3, R_0$ |
| Cycle 4 | 216 | OR R7 R3 | 212 | OR R7, R3 |
| Cycle 5 | 220 | STORE R7, $100 | 216 | STORE R7, $100 |

**Question #3 (15 points):**

A computer architecture student at AUST is working on the design of the physical memory address of the Cedar processor. The following data is present to help in the design: the processor has a byte addressable memory of 256 KBytes and a cache block size of 8 Bytes. If the student architect was to design an L1 direct-mapped cache of size 1KBytes, what should the following be:

a) The size of the memory address;
b) How is the address partitioned for the cache design. Clearly show the fields and the function of each field;
c) Which one of these fields is also stored in cache and why?
d) If the 1KByte design were to be a 2-way set associative cache, how will the address partitioning change?

**Question #4 (20 points):**

A researcher at Akkar Research Laboratories is testing the Cedar processor and its older predecessor, the Pine processor. The only distinction is that the Pine processor has NO cache memory. Both processors have a 64-bit architecture where the connection to the main memory goes through a data bus that can transfer an 8-byte word at once. To read or write one word to main memory on Pine requires 5 nanoseconds (or 20 clock cycles).

The benchmark program TestPC is utilized to evaluate both processors. This program includes two parts, a *for*-loop and the remaining code. There are 6 memory accesses in the body of the *for*-loop that repeats 200 times. In addition to these memory accesses, the benchmark requires 800 memory accesses for the remaining part of the code (for accessing instructions & data).

a) What is the total memory access time of TestPC on Pine? ( in terms of seconds and clock cycles)
b) For Cedar, we know that the cache-hit rate is 80%. When executing TestPC on Cedar, what is the total time spent on memory accesses knowing that a cache hit read requires 2 nanoseconds while the miss penalty of accessing main memory is 20 nanoseconds?
c) What is the average memory access time (AMAT) on Pine and on Cedar?
d) What were the two prime principles that triggered the use of multiple memory levels? Discuss these principles showing the distinction between them.

**Question #5 (15 points):**

You are asked to pipeline the Cedar processor and call it Oak:

1. Use at least 3 addressing modes, list, describe, and draw how each addressing mode functions.

2. Design and list the different pipeline stages used for Oak and explain what will be performed at each stage of your pipeline.

3. Schedule the following code on the Oak processor showing any stalls that are required:

Code Sequence:

| Memory Address | Memory Content |
|---|---|
| 200 | AND R2, R1 |
| 204 | ADD R4, R6 |
| 208 | SUB R3, R0 |
| 212 | OR R7, R3 |
| 216 | STORE R7, $100 |

**Question #6 (25 points):**

a) For an instruction set architecture that supports two-operand instructions, compile the following code statement – give the shortest sequence of machine instructions necessary.

$a = c * ( b + d / a );$

DIV $R_4, R_1$ 7
ADD $R_2, R_4$ 3
Mul $R_3, R_2$ 6
MOV $R_1, R_3$ 6

22 clock cycles

Assume the following:

- Variables a, b, c, & d are integers already stored in registers at locations R1, R2, R3, & R4 respectively;
- The ALU can perform the addition, division and multiplication operations;
- We are using a single-cycle CPU of 2 GHz.

b) The preceding code statement is found in a *for*-loop with an upper bound of 500 as follows:

```
for ( i = 0; i < 500; i++) {
        a = c * ( b + d / a );
}
```

When this code is compiled, the *for*-statement alone will be translated into four machine-instructions, what will be the CPU execution time for the entire *for*-loop, knowing that the CPU speed is 2 GHz?

$CPU\ time = \frac{4 + 4 \times 500}{2 \cdot 10^9} \approx 1\ \mu s$

c) Trying to evaluate performance, we move to a multi-cycle processor with a clock cycle time of 500 Pico (Pico = $10^{-12}$) seconds. For this CPU, the new instruction / cycle's (IPC) requirements are shown in the table below.

| Instruction | Clock Cycles |
|---|---|
| Addition | 3 |
| Subtraction | 3 |
| Multiplication | 6 |
| Division | 7 |
| Others | 4 |

$[(4\times6)+(3.6.7+4)500] \times 500 \times 10^{-12}$
$[24 + 11004] \times 500 \times 10^{-12}$
$= 5.5\ \mu s$

**Note:** The four machine instructions of the *for*-statement and not the for-loop body fall under the **Others** category in the table above. What is the new CPU execution time? What is the speedup over the single-cycle processor?

$\frac{1}{5.5}$

d) In order to enhance performance, we pipeline the previous multiple-cycle datapath with 5 pipeline stages. What will be the new CPU execution time if the pipeline is perfectly filled? What is the speedup over the non-pipelined multi-cycle processor?

$\approx 1.45$ $[84 + (1\times4)\ 500] 500 \times 10^{-12}$ speedup $= \frac{5}{1} = 5$

**Bonus)** If on average the CPI of pipeline stalls is 0.2, how will this affect the execution time? What is the slowdown over the pipelined processor?

$[4 \times (1.500) + 8] \times 500 \times 10^{-12}$

4 + 3 = 7