Problem 1. Beta Operation (6 points)

For the Beta instruction sequence shown below, indicate the values in the specified registers after the sequence has been executed starting at location 0. Execution terminates when the HALT() instruction is reached. Assume that all registers have been initialized to 0 before execution begins.

Remember that even though the Beta reads and writes 32-bit words from memory, all addresses are byte addresses, i.e., the addresses of successive words in memory differ by 4.

\[ . = 0 \]
\[ LD(r31, X, r0) \]
\[ CMPLE(r0, r31, r1) \]
\[ BNE(r1, L1, r1) \]
\[ ADDC(r31, 17, r2) \]
\[ BEQ(r31, L2, r31) \]
\[ L1: SRAC(r0, 4, r2) \]
\[ L2: HALT() \]

Value left in R0? \[ 0x87654321 \]
Value left in R1? \[ 0xC \]
Value left in R2? \[ 0xF8765432 \]

Value UASM assigns to symbol "L1"? \[ 0x14 \]

Give the 32-bit binary representation for the contents of location 0 (2 points):

\[ 011000000000111100011001001000 \]

LD \[ r0 \]
\[ r31 \]
\[ 0x1CE8 \]
Problem 2. Finite State Machines (6 points)

We met the DD sequential circuit in Quiz 1 – its circuit diagram is shown to the right. DD implements a FSM that has one input (IN) and one output (OUT). Please fill in the truth table and draw the state transition diagram for the FSM, remembering to label each arc with the appropriate value of IN and to specify the value of OUT for each state. Each state has been labeled with one of the four possible $S_1S_0$ values. You do not have to specify the initial state of the FSM. Since OUT is only a function of $S_1$ and $S_0$, this is a Moore machine, so the value of OUT is specified inside the state circles.

Fill in truth table and draw state transition diagram for FSM DD

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
S_1 & S_0 & IN & S_1' & S_0' & OUT \\
\hline
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 \\
\hline
\end{array}
\]

---

*Scratch diagram and truth table can be found on back of previous page*
Problem 3. Pipelining (6 points)

For each of the questions below, please create a valid K-stage pipeline of the given circuit. Each component in the circuit is annotated with its propagation delay \( t_{PD} \). Show your pipelining contours and place large black circles (●) on the signal arrows to indicate the placement of pipeline registers. Give the latency and throughput of each design, assuming ideal registers \((t_{PD}=0, t_{SETUP}=0)\). Remember that our convention is to place a pipeline register on each output.

(A) (2 points) Show the maximum-throughput 2-stage pipeline using a minimal number of registers.

\[ K = 2 \]
\[ t_{CLK} = 6 \]

Latency (ns): \( \frac{12}{3} \)

Throughput (ns\(^{-1}\)): \( \frac{1}{6} \)

(B) (2 points) Show the maximum-throughput pipeline using a minimal number of registers.

\[ K = 3 \]
\[ t_{CLK} = 4 \]

Latency (ns): \( \frac{12}{4} \)

Throughput (ns\(^{-1}\)): \( \frac{1}{4} \)

(C) (2 points) Show the maximum-throughput pipeline using a minimal number of registers. In this diagram the “4” component has been replaced by an equivalent pipelined component having 2 stages, each with a max \( t_{PD} \) of 2.

\[ K = 4 \]
\[ t_{CLK} = 3 \]

Latency (ns): \( \frac{12}{3} \)

Throughput (ns\(^{-1}\)): \( \frac{1}{3} \)

*pipelined component has two internal pipeline registers

* Scratch pipeline diagrams can be found on back of previous page
Problem 4: Stacks and Procedure Calls (12 points)

The following C program has two mutually recursive procedures \( f \) and \( g \). The corresponding assembly language is shown to the right with the code for \( f \) followed by the code for \( g \). Also shown is a partial stack trace taken when the Beta has been halted at the instruction labeled “g1:” sometime during the execution of \( f(5) \). The memory location pointed to by BP is indicated on the trace.

\[
\text{int } f(\text{int } x) \{ \\
\quad \text{return } g(x-1)+2; \\
\}
\]

\[
\text{int } g(\text{int } y) \{ \\
\quad \text{if } (y == 0) \text{return } 0; \\
\quad \text{else return } f(y)+1; \\
\}
\]

(A) (3 Points) What are the values in R0, BP and LP at the time execution was halted? Please express the values in hex or write “CAN’T TELL”.

- Value in R0: 0x \[ f(1)+1 \]
- Value in BP: 0x \[ FC \]
- Value in LP: 0x \[ 8C \]

(B) (5 Points) Please fill in the values for the blank locations in the stack trace shown on the right. Please express the values in hex.

(C) (2 Points) If the DEALLOCATE(1) instruction in the code for \( f \) is removed, will the program still work correctly, i.e., compute the same answer and obey our contract for procedure calls?

- Can remove DEALLOCATE(1) in \( f \): **YES**
- MOVE(BP,SP) will remove arg from stack

(D) (2 Points) What is the value the assembler assigns to the symbol “g” during assembly? Please express the value in hex or write “CAN’T TELL”.

- Value assembler assigns to symbol “g”: 0x \[ 64 \]