Problem 1. Beta ISA (6 Points)

Each of the following programs is loaded into a Beta’s main memory starting at location 0 and execution started with the Beta’s PC set to 0. Assume that all registers have been initialized to 0 before execution begins. Please determine the specified values after the execution reaches the HALT() instruction and the Beta stops. Write “CAN’T TELL” if the value cannot be determined. Please write all values in hex.

(A) (2 points)

```
. = 0
LD(R31, Z, R1)
SHRC(R1, 26, R1)
Z: CMPLTC(R1, 0x3C, R2)
HALT()
```

Value left in R1: 0x__________

Value left in R2: 0x__________

(B) (4 points)

```
. = 0
LD(R31, X, R0)
CMOVE(0, R1)
L: ADDC(R1, 1, R1)
SHRC(R0, 1, R0)
BNE(R0, L, R2)
HALT()
```

```
. = 0x100
X: LONG(5)
```

Value left in R0: 0x__________

Value left in R1: 0x__________

Value left in R2: 0x__________

Value assembler assigns to symbol X: 0x__________
Problem 2. Finite State Machines (5 Points)

Consider the 1-input, 1-output finite state machine with the state transition diagram shown below. Note that the single output P only depends on the current state of the FSM.

(A) (1 Point) The FSM has been processing inputs for a while and we would like to determine its current state. After entering three additional inputs “000”, we observe that we have reached a state where P=0. Please circle the possible values for the state before the additional three inputs were entered.

Possible values for state before:  S1  S2  S3  S4  S5

(B) (2 Points) Assume that the states are represented by the 3-bit binary values given on the left below. Please fill in the appropriate entries for the partial truth table shown on the right where S is the current state, I is the input value, S’ is the next state, P is the output value.

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>001</td>
</tr>
<tr>
<td>S2</td>
<td>010</td>
</tr>
<tr>
<td>S3</td>
<td>011</td>
</tr>
<tr>
<td>S4</td>
<td>100</td>
</tr>
<tr>
<td>S5</td>
<td>101</td>
</tr>
</tbody>
</table>

Fill in partial truth table

<table>
<thead>
<tr>
<th>S</th>
<th>I</th>
<th>S’</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>011</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>011</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(C) (2 points) Please identify which, if any, states are equivalent. For example, if states S1, S2, and S4 are equivalent, please write “(S1,S2,S4)”. You may need multiple parenthesized lists if more than one set of states is equivalent.

Equivalent states: _____________________________
Problem 3. Performance Measures (7 Points)

A simple combinational circuit is to be pipelined for **maximum throughput using a minimal number of registers**. For each of the questions below, please create a valid K-stage pipeline. **Show your pipelining contours** and place large black circles (●) on the signal arrows to **indicate the placement of ideal pipeline registers** (t_{PD}=0, t_{SETUP}=0). Give the latency and throughput for each design. Remember that our convention is to place a pipeline register on each output.

(A) (1 point). Show the maximum-throughput 1-stage pipeline.

Latency (ns): __________  
Throughput (ns⁻¹): __________

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

Latency (ns): __________  
Throughput (ns⁻¹): __________

(C) (2 points). Show the maximum-throughput 3-stage pipeline using a minimal number of registers.

Latency (ns): __________  
Throughput (ns⁻¹): __________

(D) (2 points). Show the maximum-throughput 4-stage pipeline using a minimal number of registers.

Latency (ns): __________  
Throughput (ns⁻¹): __________
Problem 4. Procedures & Stacks (12 Points)

The following C program implements a function (xyz) of two arguments, which returns an integer result. The assembly code for the procedure is shown on the right.

```c
int xyz(unsigned v, int b) {
    if (v == 0) return b;
    else return ???;
}
```

The execution of the procedure call `xyz(0xDECAF,0)` has been suspended just as the Beta is about to execute the instruction labeled “rtn:” during one of the recursive calls to `xyz`. A partial trace of the stack at the time execution was suspended is shown to the right below.

(A) (2 Points) Examining the assembly language for `xyz`, what is the appropriate C code for `???` in the C representation for `xyz`? Note that the Beta instructions SHR and SHRC correspond to the >> right-shift operator in C when the operands are unsigned integers.

C code for `???`: _____________________________________

(B) (5 Points) Please fill in the values for the blank locations in the stack dump shown on the right. Express the values in hex or write “---“ if value can’t be determined.

Fill in the blank locations with values (in hex!) or “---“

(C) (5 Points) Determine the specified values at the time execution was suspended. Please express each value in hex or write “CAN’T TELL” if the value cannot be determined.

Value in R1 or “CANT TELL”: 0x___________

Value in BP or “CANT TELL”: 0x___________

Value in LP or “CANT TELL”: 0x___________

Value in SP or “CANT TELL”: 0x___________

Value in PC or “CANT TELL”: 0x___________

END OF QUIZ 2!