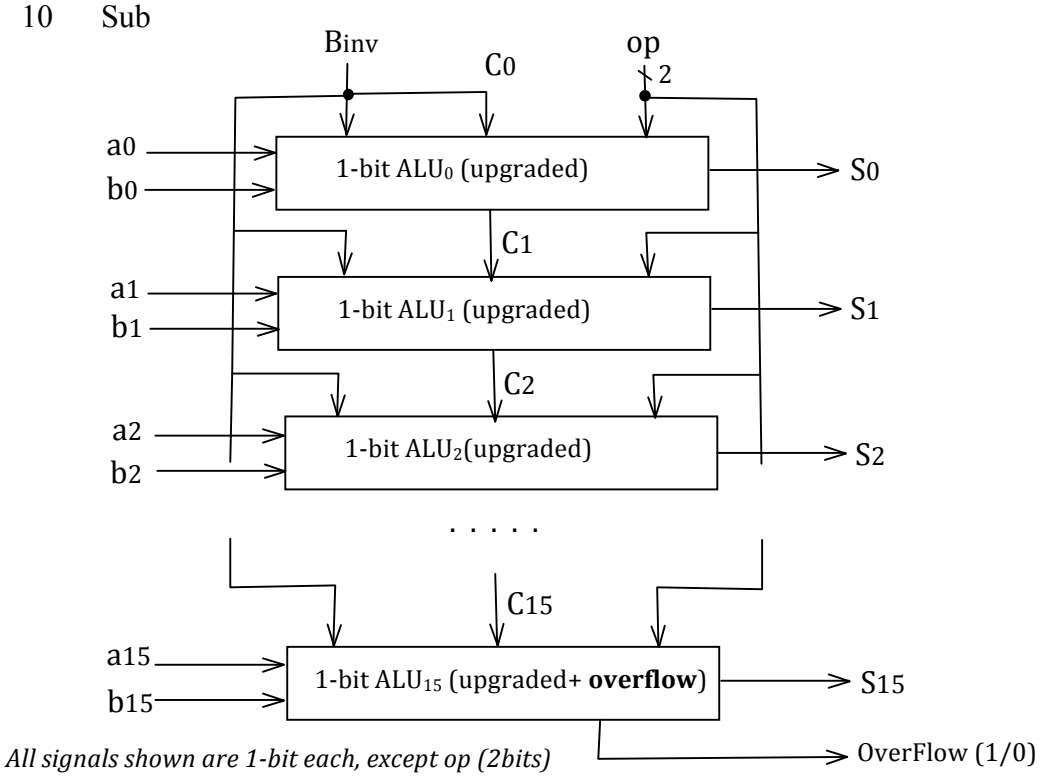
CSCI 112 (Spring 2015) Simulation Programming 2 30 pts Due: 05/01(F)

16-bit ALU simulation Phase2

Extend your 1-bit ALU simulator (80x86 assembly code) for 16-bit operations.
In this phase, add subtraction functionality in the 1-bit ALU and include overflow checking logic in the last (15^{th}) 1-bit ALU.
For the 16-bit operations, ripple-carry adder is used, i.e., 1-bit ALUs are connected by carry signals.

As shown in the diagram, the 16-bit ALU is controlled by 2 signals, which are Binv (1bit) and op (2 bits), i.e.,

Binv	op	operation
0	00	AND
0	01	OR
0	10	Add
1	10	Sub



In this phase, define one outer module (procedure) for the 16-bit-ALU, which calls 1-bit ALU as a sub-component. The main driver should prompt and access two 16-bit input numbers and control signals (combination of Binv and op – 3 bits); call the 16-bit-ALU by passing inputs as parameters; and display the inputs and outputs (16-bit result and 1bit OF signal) appropriately.

Submission:

Include good documentations in the code (global doc. and each procedure head doc.) and

- submit hard copies of the source code and run time output.
- Also send your source code by Email to jpark@csufresno.edu

Test run data are shown in the next page.

Test run data (input values) – please use these values for the output submission:

a: 0003h
b: 0004h
Binv+op: 000 (AND)

a: FFB7h
b: 0046h
Binv+op: 000 (AND)
---

a: 0003h
b: 0004h
Binv+op: 001 (OR)

a: FFB7h
b: 0046h
Binv+op: 001 (OR)
---

a: 0003h
b: 0004h
Binv+op: 010 (ADD)

a: FFB7h
b: 0046h
Binv+op: 010 (ADD)

a: FFF8h
b: FFF8h
Binv+op: 010 (ADD)
---

a: 0003h
b: 0004h
Binv+op: 110 (SUB)

a: FFB7h
b: 0046h
Binv+op: 110 (SUB)

a: FFF8h
b: FFF8h
Binv+op: 110 (SUB)