**Lab 1C Report**

1)

a)

i) The SAS (Simple Architecture System) ISA is based on the first word of its name, Simple. It is meant to do its job quickly and be completely straightforward. The ISA is designed completely around the task it is meant to do. Namely, square numbers and return bit widths. The ISA has not changed in any major way. Only branches were changed to compensate for the auto-incrementing program counter.

ii) The format of the ISA is a 2-bit Op-Code, followed by a 3-bit Description code, ending with a 3-bit exact instruction code. The Op-Code groups similar instructions together, while the description code does the same thing on a more detailed level.

iii) The ISA utilizes four general-purpose registers. Register 3 is different in that it also has a carry flag.

iv) The data memory for this ISA is 64 bytes. Therefore, addressing only requires 6 bits. The ISA does not use any general-purpose data read instructions. Any reads from memory are hard-coded, with the exception of ld5, which loads data using a pointer value in register 1.

v) The only type of branch supported in this ISA are hard-coded branches, made specifically for the programs to be run. No addresses are calculated, only the PC is changed. The branches range from -10 to +4.

b) The following is commenting of the assembly code, which reveals the flow of the program

Square:

1. Set R1 to 0
2. Set R2 to 0
3. Set R3 to 0
4. Set R4 to 0
5. Load R1 with the value at mem. loc. 0 (keep count of number of additions)
6. Load R2 with the value at mem. loc. 0 (add to R3 every loop)
7. Load R3 with the value at mem. loc. 0 (the low byte)
8. Add R2 to R3
9. Check if carry flag is set, add 1 to R4 (the high byte)
10. Decrement R1 by one
11. Check if R1 is one. If one, do not branch. Else, branch back to line 8
12. Store R4 to mem. loc. 1
13. Store R3 to mem. loc. 2
14. Halt

Width:

1. Set R1 to 0
2. Set R2 to 0
3. Set R3 to 0
4. Set R4 to 0
5. Load R1 with 31
6. Increment R1 by one
7. Load R2 with mem. Loc. [R1]
8. Store width of R2 in R3
9. Store R3 to mem. Loc. 3
10. Store R2 to mem. Loc. 4
11. Store R1 to mem. Loc. 5
12. Load R4 with R3
13. Increment R1 by one
14. Load R2 with mem. loc. [R1]
15. Store width of R2 in R3
16. If R3 is greater than R4, do not branch, else branch to line 21
17. Store R3 to mem. loc. 3 (These next few lines are only reached if a wider number is found)
18. Store R2 to mem. loc. 4
19. Store R1 to mem. loc. 5
20. Load R4 with R3
21. If R1 = 63 or if R4 = 8, branch to line 23, else do not branch
22. Branch to line 15
23. Halt program.

c)

Dynamic Instruction Number

Data Set 1:

P1: 685 P2: 187

Data Set 2:

P1: 69 P2: 201

Data Set 3:

P1: 1025 P2: 178

The rest of the information is available in the attached files.