SAMPLE.S MARKUP!

* NOTICE WE HAVE 4 FUNCTIONS: MAIN, F, G, H
* IN MAIN, IT FEELS LIKE WE HAVE 4 "LEVELS" L0-L3 → INDICATES NESTING OR FUNCTION CALLS?

```
        .globl  Xmain
Xmain:
        pushq   %rbp                    } PROLOGUE
        movq    %rsp, %rbp              }
        movl    $1, %eax                }
        pushq   %rax                    }
        movl    $2, %eax                } 3 NEW VARIABLES
        pushq   %rax                    } t, y, z INITIALIZED
        movl    $2, %eax                }
        pushq   %rax                    }
        jmp     l1
l0:
        movl    $0, %eax                } new var! z=0
        pushq   %rax                    }
        jmp     l3
l2:
        movl    -16(%rbp), %edi  → y    } THESE WILL BE ARGS SINCE CALL Xf IS NEXT
        movl    -32(%rbp), %esi  → z    }
        call    Xf
        movl    %eax, -16(%rbp)
        movl    -32(%rbp), %edi
        call    Xg
        movq    %rax, %rdi
        call    Xh
        movq    %rax, %rdi
        call    Xprint
        movl    $1, %eax                } → INCREMENT Z HERE
        movl    -32(%rbp), %edi         }
        addl    %edi, %eax              }
        movl    %eax, -32(%rbp)         }
l3:
        movl    -8(%rbp), %eax  → t
        movl    -32(%rbp), %edi → z
        cmpl    %eax, %edi              → while (z < t)
        jl      l2
        movl    -16(%rbp), %edi
        call    Xprint
        movl    -16(%rbp), %eax
        movl    -24(%rbp), %edi
        imull   %edi, %eax
        movl    %eax, -16(%rbp)
        movl    -8(%rbp), %eax
        movl    $1, %edi
        addl    %edi, %eax
        movl    %eax, -8(%rbp)
        addq    $8, %rsp
l1:
        movl    $10, %eax               } RECOGNIZE A
        movl    -8(%rbp), %edi          } WHILE LOOP!
        cmpl    %eax, %edi              }
        jl      l0
        movl    -8(%rbp), %edi
        call    Xprint
        movq    %rbp, %rsp              } EPILOGUE
        popq    %rbp                    }
        ret

        .globl  Xf
Xf:
        pushq   %rbp                    } PROLOGUE
        movq    %rsp, %rbp              }

        movl    %edi, %eax
        movl    %esi, %ecx
```

TRICKY PART HERE, TOOK SOME TRIAL + ERROR, BUT SINCE THE INSTR. ARE SO COMPACT IT MIGHT GIVE US A CLUE OF NESTING.

INNER WHILE LOOP

WHEN THE INNER LOOP EXITS, PRINT y, THEN y = y*z, THEN t = t+1

while t<10 LOOP

THESE INSTRUCTIONS WILL OCCUR AFTER LOOP, PRINT t

THESE ARE TYPICALLY ARGUMENTS

```
        subl    %ecx, %eax              } SUB THE ARGS AND SAVE IN NEW VAR C.
        pushq   %rax                    }
        movl    -8(%rbp), %eax
        pushq   %rax
        pushq   %rsi
        pushq   %rdi
        movl    -8(%rbp), %edi          } THIS IS HOW C BECAUSE WE'RE IN A NEW FUNCTION
        call    Xg
        movq    %rax, %rcx              } CALL G WITH C AS AN ARG, MULT BY C AND SAVE IN C.
        popq    %rdi
        popq    %rsi
        popq    %rax
        imull   %ecx, %eax
        movl    %eax, -8(%rbp)
        movl    -8(%rbp), %eax
        movq    %rbp, %rsp              } EPILOGUE
        popq    %rbp                    }
        ret

        .globl  Xg
Xg:
        pushq   %rbp                    } PROLOGUE
        movq    %rsp, %rbp              }
        movl    $2, %eax
        movl    %edi, %esi  ← ARG       } TRICKY → 2 * ARG - ARG
        imull   %esi, %eax              }
        movl    %edi, %esi  ← ARG       }
        subl    %esi, %eax              }
        movq    %rbp, %rsp              } EPILOGUE
        popq    %rbp                    }
        ret

        .globl  Xh
Xh:
        pushq   %rbp                    } PROLOGUE
        movq    %rsp, %rbp              }
        movl    $2, %eax
        movl    %edi, %esi  → ARG       }
        cmpl    %eax, %esi              } → IF ARG > 2, JUMP
        jnl     l4                      }
        movl    $1, %eax                } → OTHERWISE RETURN 1
        movq    %rbp, %rsp              } EPILOGUE
        popq    %rbp                    }
        ret
l4:
        pushq   %rdi
        movl    -8(%rbp), %edi
        movl    $1, %esi
        subl    %esi, %edi
        call    Xh
        popq    %rdi
        pushq   %rax
        pushq   %rdi
        movl    -16(%rbp), %edi
        movl    $2, %esi
        subl    %esi, %edi
        call    Xh
        movq    %rax, %rsi
        popq    %rdi
        popq    %rax
        addl    %esi, %eax
        movq    %rbp, %rsp              } ADDITIONAL RETURN / EPILOGUE
        popq    %rbp                    }
        ret
```

SUB 1 FROM ARG, THEN CALL H ON IT

SUB 2 FROM ARG, THEN CALL H ON IT

ADD THESE AND RETURN!

NOTICE 2 CALLS TO H, NO JUMPS OR LOOPS OR NEW VARS. HINT THAT IT COULD BE A RECURSIVE RETN

** CLUE THAT THERE WILL BE TWO RETN'S HERE