

1-1/

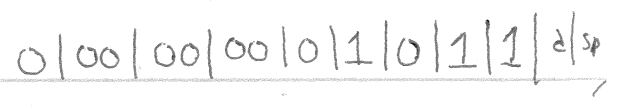
fib.mc — microcode for Fibonacci

(85 cycles)

{ ABR=0 }

START:

mar := sp; d := 1 + sp; wr;



{ D = nextaddr (sp+1) }

f := 1 + 1; wr;

{ F=2; D=sp+1; M[sp] = Fib(0) }

mbr := 1; mar := d; wr; b := 1;

{ F=2, D=sp+1, M[sp]=Fib(0), B=Fib(1) } 000100011011 b d '1' 0

e := f + 1; wr;

{ F=2, E=3, D=sp+1, M[sp]=A=Fib(0); M[sp+1]=B=Fib(1) }

LOOP:

{ Invariant: A=Fib(n) B=Fib(n+1) } SP = SP₀ + n; }

a := b + a;

{ A=Fib(n+2) B=Fib(n+1) }

d := sp + f;

mar := d; mbr := a; wr; if n then goto DONE;

{ in 1 cycle, M[sp+2] = Fib(n+2) }

sp := e + sp; wr;

{ M[sp-1] = Fib(n+2) }

b := a + b;

{ A=Fib(n+2) B=Fib(n+3) }

mar := sp; mbr := b; wr; if n then goto DONE;

wr; goto LOOP; sp := d;

DONE:

wr; rd;

VLIW - Very Long Instruction Word

q-2 / Micro Program :=
 | NL MP
 | LABEL ";" NL MP
 | Instruction NL MP

Instruction :=
 | Component ";" Instruction

Component := "mar :=" BExpr — set MAR, set B
 | "mbr :=" ShExpr — set MBR, set SH, ALU, A, B, AMUX
 | Reg " := " ShExpr — set ENC, C,
 | "alu :=" ALUExpr — set ALU, A, B, AMUX
 | "wr" — set WR
 | "rd" — set RD
 | "goto" LABEL — set COND=11, ADDR=LABEL
 | "if" Cond "then goto" LABEL — set COND = 1/z, ADDR=LABEL

Cond := N(0) | Z(10)

ShExpr := ALUExpr — set SH=00, set ALU, A, B, AMUX
 | "lshift(" ALUExpr ")" — SH=10 "
 | "rshift(" ALUExpr ")" — SH=01 "

ALUExpr := Aexpr "+" BExpr — ALU=00, set A, AMUX, B
 | AExpr — ALU=10, set A, AMUX
 | "inv(" Aexpr ")" — ALU=11, set A, AMUX
 | "band(" Aexpr ")" Bexpr ")" — ALU=01, set A, AMUX, B

AExpr := Register — A=reg AMUX=0
 | "mbr" — AMUX=1

BExpr := Register — B=reg

illegal : mar = 1 ; ac := ac + ac ; } = MAR=1, ENC=1, AMUX=0, A=AC, B=AC, C=AC
 B=1