Assignment 1

- Owen Monus
- Feb 3, 2024
- 200482797

1. You are to write an IAS program to compute the results of the following equation. Y = 1+2+...+N

a) Use the equation Y = N(N+1)/2 when writing the IAS program.

```
// assume n is at 001
// assume y is at 003
// assume 1 is at 00A
// assume 2 is at 00B

LOAD M(001)     // load n into ac
ADD M(00A)      // add 1 to contents of AC, store in AC
STOR M(002)     // store n+1 at 002
LOAD MQ,M(002)  // load n+1 into MC
MUL M(001)      // multiply, put least significant bits in MQ
LOAD MQ         // load MQ into AC. Assume n(n+1) is not large enough to
need two registers
DIV M(00B)      // divide AC by contents of 00B
LOAD MQ         // move MQ to AC since 1+2+...+N will not have a remainder
if N is a natural number
STOR M(003)     // move AC to 003
```

b) Do it the “hard way,” without using the equation from part (a).

Here is the C++ code I used as a guideline.

```
int n;
int y = 0;
int i = n;

summationLoop:
    y = y + i;
    i = i - 1;
    if (i >= 0) {
        goto summationLoop;
    }

// summation is now stored in y
```

Here is the corresponding IAS code

```
// assume 0 is at 000
// assume 1 is at 001
// assume n is at 002
// assume y is at 003
// assume the program starts at 08A

// 08A
LOAD M(000)     // Load 0 into AC
STOR M(003)     // Set y to 0

// 08B
LOAD M(002)     // Load contents of n into AC
STOR M(00A)     // store AC at 00A. This will be our iterator

// 08C
LOAD M(003)     // Load y into AC
ADD M(00A)      // Add iterator to AC, store in AC

// 08D
STOR M(003)     // store sum at 003 (y's memory address)
LOAD M(00A)     // Load iterator into AC

// 08E
SUB M(001)      // Subtract one from iterator, store in AC
STOR M(00A)     // store difference at 00A (iterator's memory address)

// 08F
JUMP + M(08C, 0:19)
// Summation is now at 003. Program continues.
```

2. Given the memory contents of the IAS computer shown below,

Address	Contents
08A	010FA210FB
08B	010FA0F08D
08C	020FA210FB

a) Show the assembly language code for the program, starting at address 08A

Instruction Address	Left Opcode	Left Address	Right opcode	Right address
08A	00000001	000011111010	00100001	000011111011
08B	00000001	000011111010	00001111	000010001101
08C	00000010	000011111010	00100001	000011111011

Address	Contents
08A	LOAD M(FA) STOR M(FB)
08B	LOAD M(FA) JUMP + M(8D, 0 : 19)
08C	LOAD –M(FA) STOR M(FB)

b) Briefly explain what this program does.

Store value of FA in FB. If the value stored in FA is non-negative, jump to left instruction in 8D. If the value stored in FA is negative, store its non-negative value at FB

- LOAD M(FA) // transfer M(X) to AC
- STOR M(FB) // transfer AC to M(X)
- LOAD M(FA) // transfer M(X) to AC
- JUMP + M(8D, 0 : 19) // If AC >= 0, take next instruction from left half of M(X)
- LOAD –M(FA) // Transfer –M(X) to the accumulator
- STOR M(FB) // transfer AC to M(X)

3.

- Pipelining: The processor overlaps operations by moving data or instructions into a conceptual pipe with all stages of the pipe processing simultaneously.
- Branch prediction: The processor looks ahead in the instruction code fetched from memory and predicts which branches, or groups of instructions, are likely to be processed next.
- Superscalar execution: This is the ability to issue more than one instruction in every processor clock cycle. Essentially multiple parallel pipelines are used.
- Multicore systems: a processor that has more than one core in one silicon wafer
- RC delay: As components on the chip decrease in size, the wire interconnects become thinner, increasing resistance. Also, the wires are closer together, increasing capacitance. This, in turn, increases delay.