Tristan Mclennan CS270 – Prof. Potter

Assignment 1

**Part 1:**

1.13.1: FP reduced by 20%:

n = 70s/(70s\*0.2) + 180s = 185s

Performance increase = 185s/250s \*100% = 7.4%

1.13.3: Total time reduced 20%: 250s – (250s\*0.2) = 200s

200s = 40s/n + 210s - Not possible. Because the entire allotment of brance time amounts to less than 20% of total time, no amount of improvement can reduce the total time to 200s.

1.15:

|  |  |  |  |
| --- | --- | --- | --- |
| P: | Et = T/P + 4: | xSpeed = T/Et: | % of Total: |
|  |  |  |  |
| 2 | 54s | 1.85 | 92.5% |
| 4 | 29s | 3.45 | 86.2% |
| 8 | 16.5s | 6.06 | 75.8% |
| 16 | 10.25s | 9.75 | 60.9% |
| 32 | 7.125s | 14.0 | 43.8% |
| 64 | 5.563s | 17.8 | 28.1% |
| 128 | 4.806s | 20.8 | 16.8% |

3) 01010101 🡪 0000000001010101 Hex: 0x0055

4.1) 11111110: Hex: 0x00FE

Decimal (unsigned): 254

Decimal (signed): -2

Base 5: 2004

4.2) 11111110 🡪 1111111111111110 Hex: 0xFFFE

**Part 2:** Op-code: Instruction: RTL:

0) Halt Halt

// Immediately halts the program

1) Load MAR (15:10) 🡨 0

MAR (9:0) 🡨 IR (9:0)

get()

Accum 🡨 MBR

// Copy address for Instruction into MAR, get contents from memory

// and store in the Accumulator.

2) Store MAR (15:10) 🡨 0

MAR (9:0) 🡨 IR (9:0)

MBR 🡨 Accum

put()

// Copy address from Instruction into MAR, move the contents of

// Accumulator into that location in memory.

3) Addc Accum 🡨 CTR + Accum

// Adds the contents of the counter register to the Accumulator,

// stores in Accumulator.

4) MVAC CTR 🡨 Accum

// Moves the contents of the Accumulator into the counter.

5) JEQ if (CTR == 0)

{

PC (15:10) 🡨 0

PC (9:0) 🡨 IR (9:0)

}

// If the counter register is zero, sets the PC to the address found

// in the instruction.

6) JLT if (CTR < 0)

{

PC (15:10) 🡨 0

PC (9:0) 🡨 IR (9:0)

}

// If the counter register is less than zero, sets the PC to the address

// found in the instruction.

7) JMP PC (15:10) 🡨 0

PC (9:0) 🡨 IR (9:0)

// Sets the PC to the address found in the instruction.

8) ADD MAR (15:10) 🡨 0

MAR (9:0) 🡨 IR (9:0)

get()

Accum 🡨 MBR + Accum

// Extracts address from instruction, loads the contents of

// that memory address into MBR, and saves the contents

// plus the Accumulator into the Accumulator.

9) SUB MAR (15:10) 🡨 0

MAR (9:0) 🡨 IR (9:0)

get()

Accum 🡨 MBR – Accum

// Extracts address from instruction, loads the contents of

// that memory address into MBR, and saves the contents

// minus the Accumulator into the Accumulator.

0xA) DEC CTR 🡨 CTR - 1

// This subtracts one from the counter register.

0xB) LA Accum (15:10) 🡨 0

Accum (9:0) 🡨 IR (9:0)

// Copies address from instruction into the Accumulator

0xC) LIA Accum (15:10) 🡨 0

Accum (9:0) 🡨 Areg (9:0)

MAR (15:10) 🡨 0

MAR (9:0) 🡨 Accum (9:0)

get()

Accum 🡨 MBR

// Copies address from Areg into Accumulator, then into MAR.

// Extracts data from that memory address, puts in Accumulator.

0xD) SIA MBR 🡨 Accum

Accum (15:10) 🡨 0

Accum (9:0) 🡨 Areg (9:0)

MAR (15:10) 🡨 0

MAR (9:0) 🡨 Accum (9:0)

put()

// Moves contents of Accumulator into MBR, then copies address

// from Areg into Accumulator, into MAR. Puts Original contents

// into memory at that address.

0xE) MVAA Areg 🡨 Accum

// Moves the Accumulator into the Areg.