

Question 4 (25 points): The main competitor to *TinyProc Inc.* is *Sneak Inc.* and they are trying to gain market share with their S75 processor. S75 uses the same Instruction Set Architecture as the TP500 and it appears to be an improved copy of the TP500. *TinyProc Inc.* has managed to gain access to an S75. The most important application in this market is **ControLux**, an application that controls the dash board of a luxury car. The compiler developed by *TinyProc Inc.* is *TinyComp* and the code generated by *TinyComp* can run both on TP500 and on S75. *Sneak Inc.* have developed their own compiler called *SneakComp* which generates code that can also run on both the TP500 and the S75. A performance comparison between TP500 and S75 reveals that S75 is twice as fast as TP500 when running **ControLux**. Management is very worried and has asked you, as the Industrial Internship Program (IIP) student with a placement at *TinyProc Inc.*, to investigate what *Sneak Inc.* has done to deliver this performance gain. Here is the information that you have to work with:

- **ControLux** compiled with *SneakComp* runs twice as fast in S75 compared with **ControLux** compiled with *TinyComp* running on TP500.
- S75 runs at 750 MHz while TP500 runs at 500 MHz.
- The instructions in the ISA of TP500 can be divided into three classes according to the number of clocks that each instruction needs to execute: ALU instructions, load/store instructions and branch instructions, When **ControLux** is compiled with *TinyComp*, its instruction mix is 40% ALU, 25% load/stores, and 35% branches.
- In TP500, the ALU instructions require 1 clock cycle to execute, the load/store instructions require 5 clock cycles, and the branches require 3 cycles.
- You have been able to measure the number of clocks per instruction for ALU and branch instructions, but not for load/store instructions, in the S75. On average it takes 1 clock cycle to execute an ALU instruction and 3 clock cycles to execute a branch in the S75. To measure the number of clock cycles per load/store instruction in the S75 you run the **ControLux** code generated by the *TinyComp* on the S75. You find that the S75 is 2.025 times faster than the TP500 when executing this code.

a. (6 points) What is the CPI of **ControLux** compiled with *TinyComp* running on TP500?

$$\begin{aligned}
 CPI_{TP500} &= 0.4 \times 1 + 0.25 \times 5 + 0.35 \times 3 \\
 &= 0.4 + 1.25 + 1.05 \\
 &= 2.7 \frac{\text{clock cycles}}{\text{instruction}}
 \end{aligned}$$

b. (7 points) What is the CPI of **ControLux** compiled with *TinyComp* when running in the S75?

The code is the same for both executions, therefore the number of instructions executed, lets call it N , is also the same

$$\begin{aligned}
Time_{TP500} &= 2.025 \times Time_{S75} \\
\frac{N \times CPI_{TP500}}{Freq_{TP500}} &= 2.025 \times \frac{N \times CPI_{S75}}{Freq_{S75}} \\
\frac{2.7}{500 \text{ MHz}} &= 2.025 \times \frac{CPI_{S75}}{750 \text{ MHz}} \\
CPI_{S75} &= \frac{750 \times 2.7}{500 \times 2.025} \\
CPI_{S75} &= 2.0 \frac{\text{clock cycles}}{\text{instruction}}
\end{aligned}$$

- c. (7 points) What is the average number of clock cycles required to execute load/store instructions when executing the **ControLux** code generated with *TinyComp* on the S75?

$$\begin{aligned}
CPI_{S75} &= 0.4 \times 1 + 0.25 \times CPI_{load/store} + 0.35 \times 3 \\
2.0 &= 0.4 + 0.25 \times CPI_{load/store} + 1.05 \\
CPI_{load/store} &= \frac{2.0 - 1.35}{0.25} \\
CPI_{load/store} &= 2.6 \frac{\text{clock cycles}}{\text{instruction}}
\end{aligned}$$

- d. (5 points) *TinyProc Inc.* has managed to obtain an executable for **ControLux** that was generated by *SneakComp*. Using hardware event counters you managed to determine that in their code both the number of branch and the number of load/store instructions were reduced by 20% in comparison with the code generated with *TinyComp*. What is the percentage change in the number of ALU instructions executed by this version of **ControLux** in comparison with the code generated with *TinyComp*?

Assume that the code generated by *TinyComp* executes 1000 instructions altogether. Therefore, it executes 400 ALU, 250 load/store, and 350 branches.

The code generated by *SneakComp* executes 200 load/stores and 280 branches.

$$\begin{aligned}
Execution\ Time_{TP500} &= \frac{I_{TP500} \times CPI_{TP500}}{Freq_{TP500}} \\
Execution\ Time_{TP500} &= \frac{1000 \times 2.7}{500 \text{ MHz}} \\
Execution\ Time_{S75} &= \frac{I_{S75} \times CPI_{S75}}{Freq_{S75}} \\
Execution\ Time_{S75} &= \frac{I_{S75} \times CPI_{S75}}{750 \text{ MHz}} \\
Execution\ Time_{TP500} &= 2 \times Execution\ Time_{S75}
\end{aligned}$$

$$\begin{aligned}\frac{1000 \times 2.7}{500 \text{ MHz}} &= 2 \times \frac{I_{S75} \times \text{CPI}_{S75}}{750 \text{ MHz}} \\ I_{S75} \times \text{CPI}_{S75} &= \frac{1000 \times 2.7 \times 1.5}{2}\end{aligned}\tag{1}$$

$$I_{S75} \times \text{CPI}_{S75} = 2025 \text{ clock cycles}\tag{2}$$

$$A = \text{number of ALU instructions executed in S75}$$

$$A \times 1 + 200 \times 2.6 + 280 \times 3 = 2025$$

$$A + 1360 = 2025$$

$$A = 665 \text{ instructions}$$

$$\text{Percentage variation} = \frac{665 - 400}{400} \times 100 = 66.3\%$$

(3)

Another way to arrive at the solution:

$$\begin{aligned}\text{Execution Time}_{S75} &= \frac{I_{S75} \times \text{CPI}_{S75}}{\text{Freq}_{S75}} \\ \text{Execution Time}_{S75} &= \frac{(480 + A) \times \frac{A+1360}{480+A}}{750 \text{ MHz}} \\ \text{Execution Time}_{S75} &= \frac{A + 1360}{750 \text{ MHz}} \\ \text{Execution Time}_{TP500} &= 2 \times \text{Execution Time}_{S75} \\ \frac{1000 \times 2.7}{500 \text{ MHz}} &= 2 \times \frac{A + 1360}{750 \text{ MHz}} \\ \frac{1000 \times 2.7 \times 1.5}{2} &= A + 1360 \\ A + 1360 &= 2025 \\ A &= 665 \text{ instructions}\end{aligned}$$