

B) Kuvaa menetelmä, jolla voit pakata VLIW-käskysanoja tiukempaan välttämällä NOP-käskyjä. Millainen käskynhakurajapinta tarvitaan?

Describe a method to package VLIW instructions in a more dense way by avoiding NOP instructions. What kind of instruction fetch interface is needed?

5. A) Taulukossa II on esitetty käskyjen jakauma eräässä sovelluksessa. Kannattaako kustannusmielessä toteuttaa jakolasku yhdellä kellojaksolla toimivalla laitteistolla, jonka pinta-ala on 1 mm^2 vai ohjelmallisesti jolloin kuluu 16 käskyä jakolaskualiohjelman toteuttamiseen + 4 käskyä joka kerta kun aliohjelmaa kutsutaan? Ohjelmallisesta toteutuksesta aiheutuu pelkästään muistin pinta-alan lisääntyminen. Oletetaan että käskysanan pituus on 32 bittiä ja muistitiheys 10 kbits/mm^2 .

Look at the application statistics in Table II. Which is better from cost point of view: to implement the division in hardware consuming only 1 instruction each time and 1 mm^2 of area, or in software, consuming 16 instructions in a subroutine + 4 instructions each time to call the subroutine but no other area overhead than from the instruction memory consumption? We can assume that the instruction length is 32 bits and memory density 10 kbits/mm^2 .

B) Toisessa sovelluksessa huomaat tarvitsevasi siirtäjää (shifter). Voit sijoittaa siirtäjän kahdella vaihtoehdoisella tavalla ALU:n suhteen

- Sarjaan ALU:n kanssa, jolloin operandin esisiirto ennen varsinaista operaatiota voidaan tehdä samalla käskyllä
- Rinnan ALU:n kanssa, jolloin tarvitaan erillinen siirtokäsky

Laske suorituskyvyn näkökulmasta kumpi on parempi, jos siirtoa tarvitaan 10%:ssa ohjelman käskyjä. ALU:n viive on 13 ns ja siirtäjän 5 ns. Setup- ja hold-ajat ovat yhteensä 2 ns. Oletetaan, että viive aritmetiikan suoritusasteen läpi määrää koko prosessorin kellojakson pituuden.

In another application you find out that you need a shifter. There are two alternatives to place a shifter in the ALU

- in series with the ALU (enabling pre-shifting of an operand in a single instruction)
- in parallel with the ALU (requiring a separate shift instruction)

Figure out which alternative is better from performance point of view if pre-shifting is needed in 10% of the instructions. The delay of the ALU is 13 ns and the delay of the shifter 5 ns. The setup and hold times needed for the registers are altogether 2 ns. The delay through the arithmetic execution stage is the limiting factor for the clock cycle.