Lektionsuppgift 2023-02-15 – Pipeline

Uppgiftsbeskrivning

Nedanstående program, där en lysdiod LED1 tänds vid nedtryckning av en tryckknapp, kan antas exekveras av en processor innehållande tre pipes.

```
.CSEG
.ORG RESET_vect
   RJMP main

main:
   CALL setup
main_loop:
   IN R24, PINB
   ANDI R24, (1 << BUTTON1)
   BRNE led1_on

led1_off:
   IN R24, PORTB
   ANDI R24, ~(1 << LED1)
   OUT PORTB, R24
   RJMP main_loop
led1_on:
   IN R24, PORTB
   ORI R24, (1 << LED1)
   OUT PORTB, R24
   RJMP main_loop

setup:
   LDI R16, (1 << LED1)
   OUT DDRB, R16
   LDI R17, (1 << BUTTON1)
   OUT PORTB, R17
   RET
```

Anta att *main_loop* körs två varv, där tryckknappen BUTTON1 trycks ned efter det första varvet. Första varvet ska alltså *led1_off* köras, nästa varv ska *led1_on* köras. Totalt ska alltså 21 instruktioner exekveras.

a) Fyll i tillstånden för respektive pipeline samt utförd instruktion på nästa sida!

b) Ange verkningsgraden (ratio mellan antalet exekverade instruktioner samt antalet klockcykler) vid användning av en respektive tre pipes.

Svar: Vid användning av en pipe exekveras en instruktion var tredje klockcykel, verkningsgraden är därmed 33 %. Efter 21 instruktioner genomfördes under programmets gång hade detta därmed genomförts under 21 * 3 = 63 klockcykler.

Vid användning av tre pipes genomfördes de 21 instruktionerna under 33 klockcykler, vilket medför en verkningsgrad på 21/33 = 0.64, vilket motsvarar 64 %.

c) Ange hur mycket effektivare exekvering sker vid användning av tre pipelines i stället för en i detta program.

Svar: Genom att jämföra antalet klockcykler som krävdes för att exekvera de 21 instruktionerna i detta program ser vi att det är 63 / 33 = 1.9 gånger effektivare med tre pipes. Ju fler hoppinstruktioner som genomförs, desto mindre skillnad blir det, men runt dubbelt så effektivt kan man räkna med att det blir. Detta kan vägas mot att konstruktionen blir mycket mer komplicerad och det krävs mycket mer hårdvara.

Tabell 1: Tillstånd för respektive pipeline samt utförd instruktion varje klockcykel.

Klockcykel	Pipe1	Pipe2	Pipe3	Utförd instruktion
1	FETCH (RJMP)	-	-	-
2	DECODE	FETCH	-	-
3	EXECUTE	DECODE	FETCH	RJMP main
4	FETCH (CALL)	-	-	-
5	DECODE	FETCH	-	-
6	EXECUTE	DECODE	FETCH	CALL setup
7	FETCH (LDI)	-	-	-
8	DECODE	FETCH (OUT)	-	-
9	EXECUTE	DECODE	FETCH (LDI)	LDI R16, (1 << LED1)
10	FETCH (OUT)	EXECUTE	DECODE	OUT DDRB, R16
11	DECODE	FETCH (RET)	EXECUTE	LDI R17, (1 << BUTTON1)
12	EXECUTE	DECODE	FETCH (-)	OUT PORTB, R17
13	FETCH (-)	EXECUTE	DECODE	RET
14	FETCH (IN)	-	-	-
15	DECODE	FETCH (ANDI)	-	-
16	EXECUTE	DECODE	FETCH (BRNE)	IN R24, PINB
17	FETCH (IN)	EXECUTE	DECODE	ANDI R24, (1 << BUTTON1)
18	DECODE	FETCH (ANDI)	EXECUTE	BRNE led1_on
19	EXECUTE	DECODE	FETCH (OUT)	IN R24, PORTB
20	FETCH (RJMP)	EXECUTE	DECODE	ANDI R24,~(1 << LED1)
21	DECODE	FETCH (LDI)	EXECUTE	OUT PORTB, R24
22	EXECUTE	DECODE	FETCH (OUT)	RJMP main_loop
23	FETCH (IN)	-	-	-
24	DECODE	FETCH (ANDI)	-	-
25	EXECUTE	DECODE	FETCH (BRNE)	IN R24, PINB
26	FETCH (IN)	EXECUTE	DECODE	ANDI R24, (1 << BUTTON)
27	DECODE	FETCH (ANDI)	EXECUTE	BRNE led1_on
28	FETCH (IN)	-	-	-
29	DECODE	FETCH (ORI)	-	-
30	EXECUTE	DECODE	FETCH (OUT)	IN R24, PORTB
31	FETCH (RJMP)	EXECUTE	DECODE	ORI R24, (1 << LED1)
32	DECODE	FETCH (LDI)	EXECUTE	OUT PORTB, R24
33	EXECUTE	DECODE	FETCH (OUT)	RJMP main_loop