Tentamen i Datorsystemteknik EDA332 / DIT122

Tid och plats: Fredagen den 11 Oktober, 2019, fm, SB-salar

Examinator: Lars R Bengtsson

Institution: Data- och Informationsteknik

Förfrågningar: Lars Bengtsson (tel.8441);

Rättningsgranskning: Fredagen den 25 Oktober kl. 13.45-14.15; hos Lars på D & IT inst. plan 4.

Betygsgränser:
3: 24–35 poäng, 4: 36– 47 poäng, 5: 48–60 poäng

Tillåtna hjälpmedel:

Referenskort för MIPS-arkitekturen (”Green card”)

Av Chalmers typgodkänd kalkylator.

Allmänt: Svar lämnas på lösblad. Använd gärna figurer. Tentan kan delas för rättning; kombinera inte flera uppgifter på samma blad.

Textsvar skrives gärna på engelska – tack!

I förekommande fall gör de antaganden som behövs och motivera dessa.

Maximal poäng på varje deluppgift anges inom parentes efter uppgiftstexten.

Lycka till!

(a) Kommentera följande program rad för rad och beskriv sedan kort vad hela programmet gör. (6p)

```
        addi    $t0, $zero, 0
        addi    $t1, $zero, 0

L1:     sll     $t2, $t0, 2
        add     $t3, $a0, $t2
        lw      $t4, 0($t3)
        add     $t1, $t1, $t4
        addi    $t0, $t0,1
        slt     $t5, $t0, $a1
        bne     $t5, $zero, L1
        add     $v0, $zero, $t1
```

(b) Polling vs. avbrott.
En processor med klockfrekvensen 1 GHz behöver läsa datapaket från en viss I/O enhet. Enheten genererar 1 byte var tjugonde mikrosekund; att ta vara på denna byte tar 1000 cykler. Vi kan välja mellan att hantera dessa data genom pollning eller genom avbrottsstyrning.

i. Ett avbrott tar 200 cykler att hantera. Hur stor del av processorkapaciteten åtgår för att hantera denna I/O-enhet om avbrottsstyrning används? (3p)

ii. Antag nu istället att pollning används. 75 cykler åtgår för varje pollningsoperation. Hur stor del av processorkapaciteten krävs i detta fall, givet att svarstiden skall vara densamma som i avbrottsfallet? (3p)

2. Ett visst program exekveras i en processorsimulator liknande den ni sett i laborationer och inlämningsuppgift. Processorn har ett icke-associativt cache med totala storleken 1 KB. Det visar sig att andelen missar är 38% om blockstorleken sätts till 4B; 22% vid blockstorleken 16B; 19% för blockstorleken 64B; och 27% vid blockstorleken 256B. Miss penalty är 4+b/4 cykler, där b är blockstorleken, och det görs 1,2 minnesreferenser per instruktion. CPI utan cachemissar är 2. Klockfrekvensen antas vara oberoende av blockstorleken.

(a) Varför minskar andelen missar när blockstorleken ökar från 4B till 16B och från 16B till 64B? (4p)
(b) Varför ökar andelen missar åter när blockstorleken ökas till 256B? (4p)
(c) Vilken blockstorlek bör användas för att minimera exekveringstiden? (4p)

3. Cacheminnen kan karaktäriseras med användning av följande parametrar:

S = total lagringskapacitet i antal bytes,
B = blockstorlek i antal bytes,
A = associativitet (t.ex.. A=4 betyder 4-vägs set associativitet),
W = antal bitar i den fysiska adressen

Antag att:

- S, B, och A kan utryckas med 2-potenser, och $s = \log_2 S$, $b = \log_2 B$, $a = \log_2 A$.
- Cachen är fysiskt adresserad
- Alla adresser är byteadresser
- Random används som utbytesalgoritm
- Write-back används som uppdateringsstrategi

Utryck följande som funktioner av parametrarna S, s, B, b, A, a, W:

a. Antal block i cachen (2 p)
b. Antal Sets i cachen (2 p)
c. Antal bitar i en cachetag (2 p)
d. Totalt antal bitar som cachen måste kunna lagra (6 p)
e. Minsta sidstorlek (i antal bytes) i ett virtuellt minnessystem om en cacheuppslagning skall kunna ske parallellt med en adressöversättning i ett virtuellt minne. (6 p)

4.

a. Beskriv hur de ingående termerna i formeln för CPU-tid påverkas av olika konstruktionsval avseende: ISA, program, kompilator, processoruppbyggnad, minnesorganisation samt hårdvaruteknologi. (6 p)

b. Under vilka förutsättningar kan datakonflikter mellan instruktioner lösas med hjälp av forwarding i en pipeline? (3 p)

c. På vilket sätt skulle forwarding kunna försämra prestanda? (2 p)

d. Ge förslag på minst fyra olika förbättringar (optimeringar) så att CPU-tiden minskar hos MIPS-koden nedan. Eventuella datakonflikter löses med forwarding och vid styrkonflikter används fördröjt hopp (*delayed branch*). Antag att alla register innehåller värdet 0 från början. (4 p)

```
        addi $a1, $zero, 8
        add $t1,   $a1, $zero

L1:     add  $a2,  $t1, $t1
        addi $t0, $zero, 4
        add  $a1, $a1, $t0
        lw   $t2,  0($a1)
        add $v0, $v0, $t2
        add $v0, $v0, $v0
        add $v0, $v0, $v0
        bne $t2, $zero, L1
        nop
        add $t1, $zero, $zero
        sw  $v0, 4($a2)
```

f. Rangordna procedurerna clear1, clear2 och clear3 nedan efter grad av rumslokalitet. Motivera din rangordning. (3 p)

```
#define N 1000

typedef struct {

    int vel[3];
    int acc[3];

} point;

point p[N];
```

```
void clear1 (point *p, int n)

{
        int i, j;

        for (i = 0; i < n; i++) {
                for (j = 0; j < 3; j++)
                        p[i].vel[j] = 0;
                for (j = 0; j < 3; j++)
                        p[i].acc[j] = 0;
        }
}

void clear2(point *p, int n)

{
        int i, j;

        for (i = 0; i < n; i++) {
                for (j = 0; j < 3; j++) {
                        p[i].vel[j] = 0;
                        p[i].acc[j] = 0;
                }
        }
}

void clear3(point *p, int n)

{
        int i, j;

        for (j = 0; j < 3; j++) {
                for (i = 0; i < n; i++)
                        p[i].vel[j] = 0;
                for (i = 0; i < n; i++)
                        p[i].acc[j] = 0;
        }
}
```

SLUT