

Student:

Collegekaartnummer:

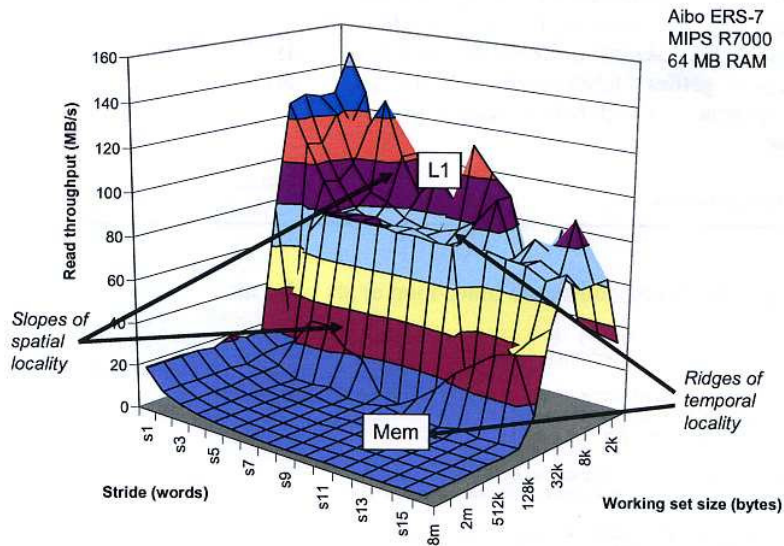
## Tentamen Computersystemen voor AI programmeurs

baiCSAI3 2e jaar bachelor AI, 2e semester 22 maart 2010

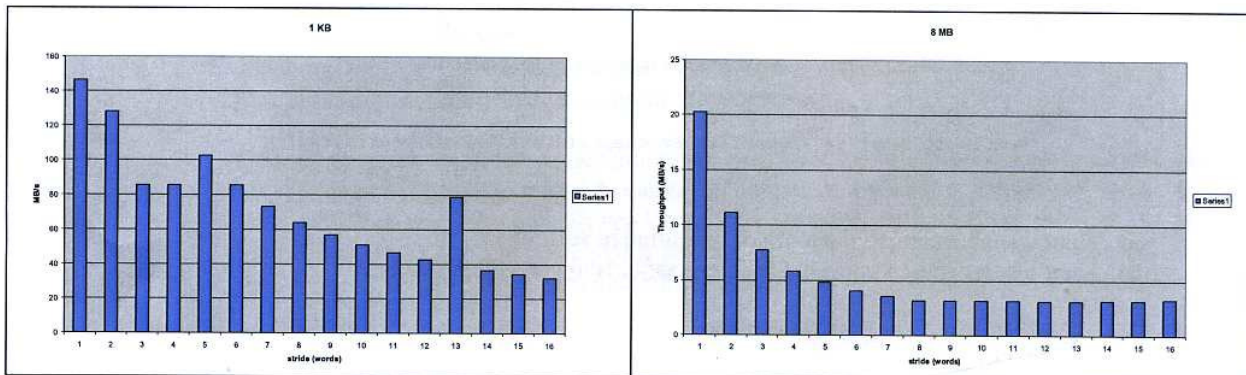
### vraag 1

U heeft een Aibo ERS-7 onder uw hoede gekregen. Deze robotohnd bevat een MIPS processor van een onbekende fabrikant.

U meet het geheugen als functie van de *spatial* en *temporal locality*. U krijgt het volgende resultaat:



In de volgende figuur zijn twee doorsneden van deze figuur te zien, voor respectievelijk de kleinste en grootste *working sets*.



- Met welke snelheid kan data uit het RAM geheugen gelezen worden?
- Met welke snelheid kan data uit de *cache* gelezen worden?
- Reken dit voor een *4-byte word* om in microseconden en tikken, als men weet dat de *pipeline clock* op 96 MHz loopt.

Student:

Collegekaartnummer:

## vraag 2

Neem het volgende algoritme uit de robotica c) in ogenschouw:

---

### Algorithm 1 Particle Filter Localization

---

```
1: for all  $p \in$  particles do
2:    $p.exponent \leftarrow 0$ 
3: end for
4: for all  $o \in$  observations do
5:   for all  $p \in$  particles do
6:      $dist_{expected} \leftarrow$  getDistanceToPoint( $p.pos, o.pos$ )
7:      $bear_{expected} \leftarrow$  getBearingToPoint( $p.pos, o.pos$ )
8:      $pe_{dis} \leftarrow$  getDistanceSimExponent( $dist_{expected}, o.dist$ )
9:      $pe_{bear} \leftarrow$  getBearingSimExponent( $bear_{expected}, o.bear$ )
10:     $p.exponent \leftarrow p.exponent + pe_{dis} + pe_{bear}$ 
11:   end for
12: end for
13: estimatePose(particles)
```

---

In de C++ code ziet de corresponderende inner-loop er als volgt uit:

```
for (int oi = 0; oi < on ; oi++) {
    Observation &obs = obsData[ob + oi];

    // the observed distance and bearing to the landmark
    float observedDistance = obs.d;
    AngRad observedBearing = obs.b;

    // location of the reference object
    Point2D refObjPos = REF_OBJ_POS_ARR[obs.id];

    for (int p = 0; p < NUM_SCALAR_PARTICLES; p++) {
        Particle &part = scalarParticles[p];

        // if we were at the current particle, this is the expected
        // distance and expected bearing to the landmark's known location
        float expectedDistance = part.getDistanceTo(refObjPos);
        AngRad expectedBearing = part.getBearingTo(refObjPos);

        float distanceExp = getDistanceSimExponent(expectedDistance,
                                                    observedDistance,
                                                    DIST_EXP_COEFF);

        float bearingExp = getBearingSimExponent(expectedBearing,
                                                  observedBearing,
                                                  BEAR_EXP_COEFF);

        scalarProb[p] += ProbabilityExponents(distanceExp, bearingExp);
    }
}
```

- Kunt u aanbevelingen doen om deze routine te versnellen?
- Kunt u beredeneren hoeveel CPE deze aanbevelingen zullen schelen?

[1] Peter Djeu, Michael Quinlan and Peter Stone, "Improving Particle Filter Performance Using SSE Instructions", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, USA, 2009.

Student:

Collegekaartnummer:

### Vraag 3

Bestudeer het volgende C programma, dat bedoelt om de gereserveerde kamers in een klein familiehotel bij te houden. Elk element in de `residents` matrix bevat de naam van een gast die deze kamer heeft gereserveerd; `FLOORS` is een constante die het aantal verdiepingen van het hotel representeert. `ROOMS` is een constante die het aantal kamers per verdieping representeert. Beiden constanten zijn elders met een `#define` gedefinieerd. `LEN`, het maximum aantal bytes gallocceerd voor de naam, is gedefinieerd op 12.

```
char residents[FLOORS][ROOMS][LEN];

void reserve_room(int floor, int room, char *custname)
{
    strcpy(residents[floor][room], custname);
}
```

De assembly code van de functie ziet er als volgt uit:

```
reserve_room:
    pushl %ebp
    movl %esp,%ebp
    movl 12(%ebp),%eax
    movl 16(%ebp),%edx
    pushl %edx
    movl 8(%ebp),%edx
    sall $4,%edx
    subl 8(%ebp),%edx
    leal (%eax,%eax,2),%eax
    leal residents(,%eax,4),%eax
    leal (%eax,%edx,4),%edx
    pushl %edx
    call strcpy
    movl %ebp,%esp
    popl %ebp
    ret
```

- Wat is de waarde van `ROOMS`?
- Ten gevolge van een vreemde bug, roept het programma `residents[0][1][-2]` aan. Welk element van de matrix `residents` wordt er eigenlijk aangeroepen? (Geef je antwoord als een combinatie van drie unsigned ints. Je mag er vanuit gaan dat zowel `FLOORS` als `ROOMS` groter zijn dan 1).
- De programmeur realiseert zich dat deze implementatie niet bepaald efficiënt met geheugen omgaat. Om het familiehotel in deze tijden van crisis toch nog concurrent te houden besluit hij het geheugen gebruik te verbeteren.

De programmeur verandert de declaratie van `residents` in een matrix van pointers, die verwijst naar het geheugen waar de naam van de gast is opgeslagen. De nieuwe code allocceert alleen nog geheugen als een kamer echt gereserveerd is. Anders verwijst `residents[f][r]` naar een NULL pointer. Ga voor deze opgave er vanuit dat de functie `malloc` geen extra geheugenruimte gebruikt voor administratieve doeleinden.

**Student:**

**Collegekaartnummer:**

De nieuwe code ziet er als volgt uit:

```
char *residents[FLOORS][ROOMS];

Void reserve_room(int floor, int room, char *custname)
{
    residents[floor][room] = malloc(LEN);
    strcpy(residents[floor][room], custname);
}
```

Na een paar maanden komt de programmeur terug om zijn besparing op te meten. In die periode had het hotel een bezettingsgraad van 20%. De programmeur ziet tot zijn genoegen dat de nieuwe routine 168 bytes minder geheugen heeft gebruikt. Hoeveel verdiepingen heeft dit hotel (in andere woorden, wat was de waarde van FLOORS)?

#### Vraag 4

De volgende code is het hart van een *tiny* webserver die is per *clients* een proces opstart:

```
listenfd = Open_listenfd(port);
while (1) {
    connfd = Accept(listenfd, (SA *) &clientaddr, &clientlen);
    if (Fork() == 0) {
        Close(listenfd); /* Child closes its listening socket */
        dotit(connfd); /* Child services client */
        Close(connfd); /* Child closes connection with client */
        exit(0); /* Child exits */
    }
    Close(connfd); /* Parent closes connected socket (important!) */
}
```

- Waarom is het belangrijk dat de *parent* de *file-descriptor* `connfd` sluit?
- Als de *parent* de *file-descriptor* `connfd` sluit, is de *child* dan nog wel in staat om met de *client* te communiceren? Verklaar uw antwoord.
- Dat de *child* de *file-descriptor* `connfd` sluit, is netjes, doch strikt niet nodig. Waarom niet?