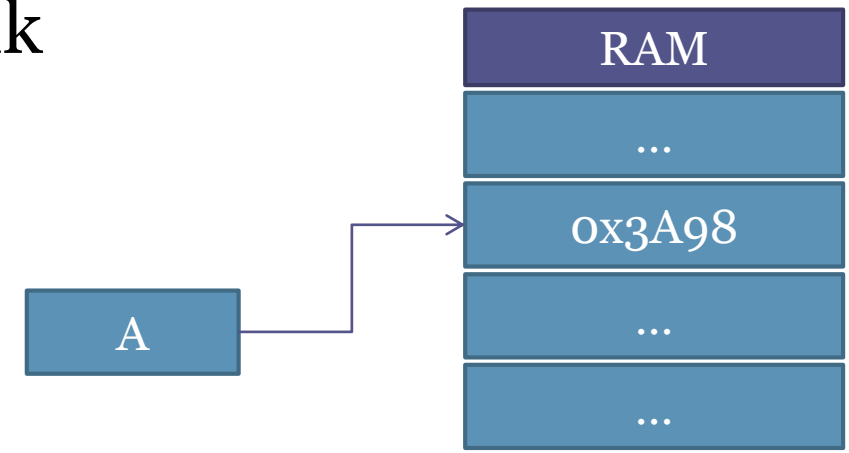
Übung: Parallele Algorithmen

Einführung in „C“ Pointer

- Zeigen auf einen Speicherbereich
- Erkennbar durch * am Variablennamen (int* a)
- Größe Abhängig von der Rechnerarchitektur
- Ein Pointer weiß seine „Inhaltsgröße“ nicht
- Arithmetik



Operatoren

& Operator

Liefert die Adresse einer im Speicher liegenden Variable

* Operator

Greift auf den Inhalt der Zeigervariable zu

Beispiele: & und * Operator

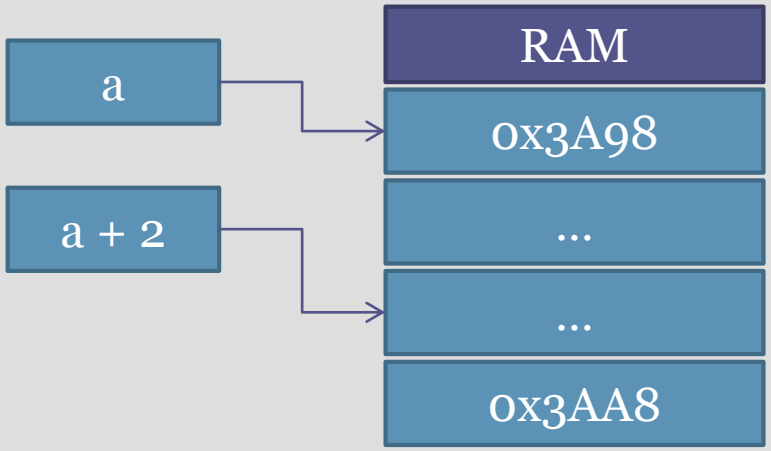
```
int size = 10;
int* sizePtr = &size;
int sizeCpy = *sizePtr;
```

Beispiele: „C“ Array und Arithmetik

```
int* a = new int[4]; FillWithZero(a, 4);
int* kp1 = a + 2;
int kp2 = *(a + 2);
int kp3 = (* a) + 2;

*(a + i) == a[i] !

~~a.length~~
```



LWJGL Array vs. „C“ Array

LWJGL	C
IntBuffer	int*
FloatBuffer	float*
…	…
PointerBuffer (kapselt einen architekturunabhängigen Pointer, z.B. CLEvent)	Keine Entsprechung

Eine Funktion im Vergleich

LWJGL

```
int clEnqueueReadBuffer ( CLCommandQueue q, CLMem
m, int blocking_read, long offset, IntBuffer ptr, ...)
```

C

```
cl_int clEnqueueReadBuffer ( cl_command_queue q, cl_mem
m, cl_bool block, size_t offset, size_t bytes_to_read,
void *ptr, ...)
```

Beispiel: LWJGL clEnqueueReadBuffer

```
CLMem devPtr <- {1,2,3,4,5}
IntBuffer hPtr <- {0,0,0,0,0}
hPtr.position(2); //Start Position
hPtr.limit(4); //End Position (2 (4-2) Byte lesen)

clEnqueueReadBuffer(queue, devPtr, 1,
3 * SIZE_OF_INT, //Offset im Device Memory (Byte)
hPtr /*Hostpointer */, …);

Achtung: Rücksetzten nicht vergessen!
hPtr.limit(hPtr.capacity());
hPtr.position(0);
```

Beispiel: „C“ clEnqueueReadBuffer

```
cl_mem devPtr <- {1,2,3,4,5}
int* hostPtr = new int[5];

clEnqueueReadBuffer(queue, devPtr, 1,
2 * sizeof(int), //Offset Device/Host Memory (Byte)
2 * sizeof(int), //Bytes zu lesen
hostPtr /*Hostpointer*/, …);
```