Example: V-bits and A-bits in Valgrind

```
1:  void main()
2:  {
3:    int a;
4:    int *p;
5:    a = 99;
6:    p = (int*) malloc(2*sizeof(int));
7:    p[0] = 0;
8:    printf("p[1]=%d\n", p[1]); //uninitialized read
9:    p[1] &= 0xFFFFFFFE;
10:   if(p[1] & 0x00000001)
11:    ;
12:   p[0] = p[1]; // copy is OK, even if not fully initialized
13:   free(p);
14:   p[0] = 0; //unallocated write
15: }
```

After line 4:

	content	V-bits	A-bits
a	random	0x00000000	0xF
p	random	0x00000000	0xF

After line 5:

	content	V-bits	A-bits
a	99	0xFFFFFFFF	0xF
p	random	0x00000000	0xF

After line 6:

	content	V-bits	A-bits
a	99	0xFFFFFFFF	0xF
p	Address of *p	0xFFFFFFFF	0xF
p[0]	random	0x00000000	0xF
p[1]	random	0x00000000	0xF

After line 7:

	content	V-bits	A-bits
a	99	0xFFFFFFFF	0xF
p	Address of *p	0xFFFFFFFF	0xF
p[0]	0	0xFFFFFFFF	0xF
p[1]	random	0x00000000	0xF

Line 8: Uses p[1] -> check a-bits and v-bits
a-bits are fine (no unallocated read)
v-bits are not (using uninitialized data) -> Warning!

After line 9:

	content	V-bits	A-bits
a	99	0xFFFFFFFF	0xF
p	Address of *p	0xFFFFFFFF	0xF
p[0]	0	0xFFFFFFFF	0xF
p[1]	Random, last bit is 0	0x00000001	0xF

Line 10: Uses the last bit of p[1] -> check a-bits and <u>this one</u> v-bit
a-bits are fine (no unallocated read)
this one v-bit is fine -> NO Warning!

Line 12: check a-bits. If we only copy data, we do NOT check the v-bits, we only copy them.
Reason: real code copies uninitialized data quite a lot.

After line 12:

	content	V-bits	A-bits
a	99	0xFFFFFFFF	0xF
p	Address of *p	0xFFFFFFFF	0xF
p[0]	Random, last bit is 0	0x00000001	0xF
p[1]	Random, last bit is 0	0x00000001	0xF

After line 13:

	content	V-bits	A-bits
a	99	0xFFFFFFFF	0xF
p	Address of *p	0xFFFFFFFF	0xF
p[0]	Random, last bit is 0	0x00000001	0x0
p[1]	Random, last bit is 0	0x00000001	0x0

Line 14: Writes p[0] -> check a-bits -> Warning