Line 1: Initialize integer

```
a: 0(sp)     c: 12(sp)
b: 4(sp)     d: 52(sp)
```

```c
int a = 5;
char b[] = "string";
int c[10];
uint8_t d = b[3];
c[4] = a+d;
c[a] = 20;
```

```
li t0 5 # R[t0] = 5
sw t0 0(sp) # store int a on stack
```

t0 `0x0000 0005`

	+3	+2	+1	+0
0(sp)	00	00	00	05
4(sp)	FA	CE	FA	CE
8(sp)	FA	CE	FA	CE
12(sp)	FA	CE	FA	CE
16(sp)	FA	CE	FA	CE

Gray: (random garbage)

Line 2: Initialize string?

```
a: 0(sp)    c: 12(sp)
b: 4(sp)    d: 52(sp)
```

```c
int a = 5;
char b[] = "string";
int c[10];
uint8_t d = b[3];
c[4] = a+d;
c[a] = 20;
```

```
li t0 0x73
sb t0 4(sp)
li t0 0x74
sb t0 5(sp)
li t0 0x72
sb t0 6(sp)
li t0 0x69
sb t0 7(sp)
li t0 0x6E
sb t0 8(sp)
li t0 0x67
sb t0 9(sp)
sb x0 10(sp)
```

	's'	't'	'r'	'i'	'n'	'g'	'\0'
ASCII (0x)	73	74	72	69	6E	67	00

	+3	+2	+1	+0
0(sp)	00	00	00	05
4(sp)	69	72	74	73
8(sp)	FA	00	67	6E
12(sp)	FA	CE	FA	CE
16(sp)	FA	CE	FA	CE

- Seems costly!
- Can we do better?

Gray: (random garbage)

UC Berkeley

Line 2: String, better

```
a: 0(sp)    c: 12(sp)
b: 4(sp)    d: 52(sp)
```

```
int a = 5;
char b[] = "string";
int c[10];
uint8_t d = b[3];
c[4] = a+d;
c[a] = 20;
```

Note little endian

t0	0x6972 7473
t1	0x0000 676E
t2	

ASCII	's'	't'	'r'	'i'	'n'	'g'	'\0'
(0x)	73	74	72	69	6E	67	00

$$\underbrace{73\quad 74\quad 72\quad 69}_{\texttt{0x697274}\underline{\texttt{73}}}\qquad \underbrace{6E\quad 67\quad 00}_{\texttt{0x}\underline{\texttt{00}}\texttt{00676E}}$$

4B integer

LSB, 's'

zero pad

	+3	+2	+1	+0
0(sp)	00	00	00	05
4(sp)	69	72	74	73
8(sp)	00	00	67	6E
12(sp)	FA	CE	FA	CE
16(sp)	FA	CE	FA	CE

```
li t0 0x69727473 # load "stri"
sw t0 4(sp) # store first part of string
li t1 0x0000676E # load rest of string
sw t1 8(sp) # store rest of string
```

UC Berkeley

Gray: (random garbage)

Line 3: Declare array

```
a: 0(sp)    c: 12(sp)
b: 4(sp)    d: 52(sp)
```

```
int a = 5;
char b[] = "string";
int c[10];
uint8_t d = b[3];
c[4] = a+d;
c[a] = 20;
```

- No data moves between registers and memory!
- **No instructions needed**—leave things as-is!

	+3	+2	+1	+0
0(sp)	00	00	00	05
4(sp)	69	72	74	73
8(sp)	00	00	67	6E
12(sp)	FA	CE	FA	CE
16(sp)	FA	CE	FA	CE

Gray: (random garbage)

UC Berkeley

Line 4: Load/store byte

```
a: 0(sp)     c: 12(sp)
b: 4(sp)     d: 52(sp)
```

```
int a = 5;
char b[] = "string";
int c[10]
uint8_t d = b[3];
c[4] = a+d;
c[a] = 20;
```

t0 `0x0000 0069`

```
lb t0 7(sp) # 4(sp) from b, 3(sp) from [3]
sb t0 52(sp)# store into d
```

	+3	+2	+1	+0
0(sp)	00	00	00	05
4(sp)	69	72	74	73
8(sp)	00	00	67	6E
…	…			
52(sp)	FA	CE	FA	69

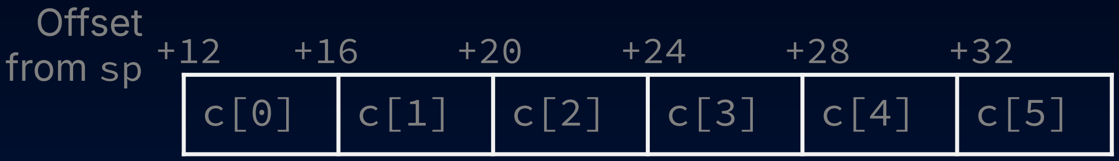
Gray: (random garbage)

Line 5: Constant array index

```
a: 0(sp)    c: 12(sp)
b: 4(sp)    d: 52(sp)
```

```
int a = 5;
char b[] = "string";
int c[10];
uint8_t d = b[3];
c[4] = a+d;
c[a] = 20;
```

t0	0x0000 0005
t1	0x0000 0069
t2	0x0000 006E



	+3	+2	+1	+0
0(sp)	00	00	00	05
4(sp)	69	72	74	73
...	...			
28(sp)	00	00	00	6E
...	...			
52(sp)	FA	CE	FA	69

- Pointer arithmetic: 4*sizeof(int) = 16
- 16 bytes after 12(sp) → 28(sp)

```
lw t0 0(sp) # load a
lbu t1 52(sp) # load d
add t2 t0 t1 # R[t2] = a+d
sw t2 28(sp) # 12(sp) from c, 16(sp) from [4]
```

UC Berkeley