CSE 2037, week 6

→ Review of Pointers

C code

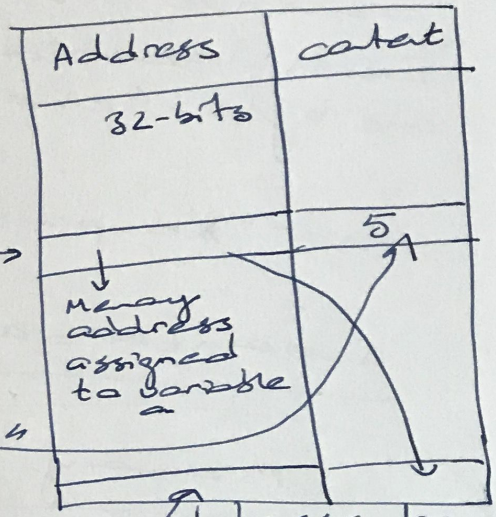
Declare a variable

```
int a;
```

within code

```
a = 5;
```

Memory



"pointer is another variable"

→ I am pointing to a memory address by the pointer

→ The variable declared as "pointer" keeps a "memory address"

```
int *pointer;
```

→ pointer variable declaration

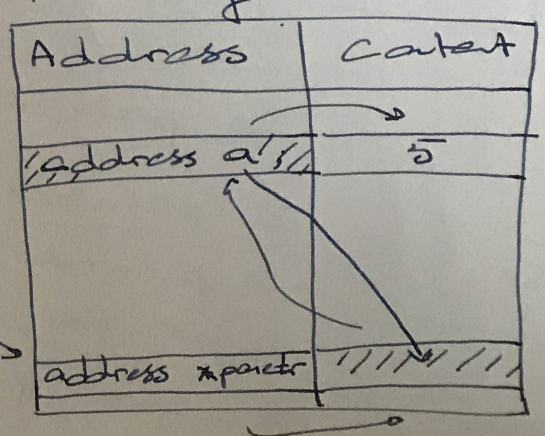
```
pointer = &a;
```

→ assigning "address" of variable a to my pointer

```
int a;
a = 5;
int *pointer;
pointer = &a;
```

*pointer

Memory



Arrays in C Language

~~char[5]~~

char a[5] = {1,2,3,4,5}

⇒ Assign 5 "consecutive" "successive" memory locations
Each location keeps one byte "char"

Memory

address	content
addr 0.	a[0] → 1
1.	a[1] → 2
1	a[2] → 3
1	a[3] → 4
1	a[4] → 5
	~~a[5]~~

char *a-pointer;

a-pointer = a;

Memory

address	content
0x11a2	a[0] = 1
0x11a3	a[1] = 2
0x11a4	a[2] = 3
0x11a5	a[3] = 4
0x11a6	a[4] = 5
0x27fc	0x11a2

char a[5]

char *a-pointer;

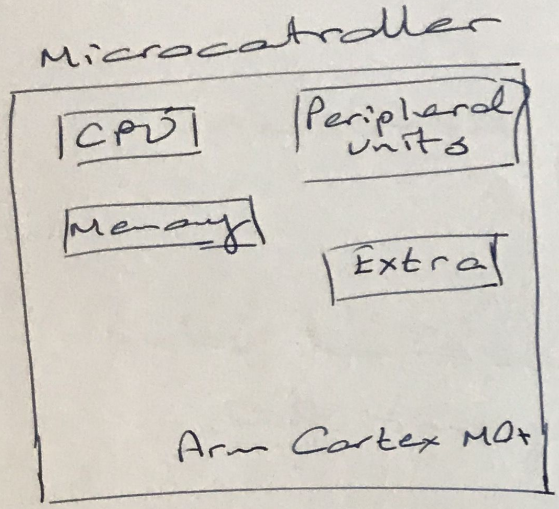
a-pointer = a;

a-pointer += 1; (a-pointer++;) 0x11a3

(a-pointer = a-pointer + 1;)

*a-pointer = 0;

Assembly Language



Assembly Language

→ Programming language like C.

→ Lowest level language

Hardware ⟵ ⟶ Software

Lowest level → I can reach all the elements of the MC.

⇒ It does not have "for", "if", "while" ...

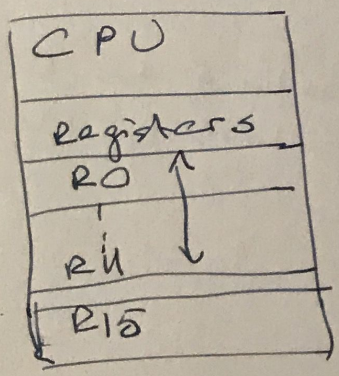
You do not "assign" a variable

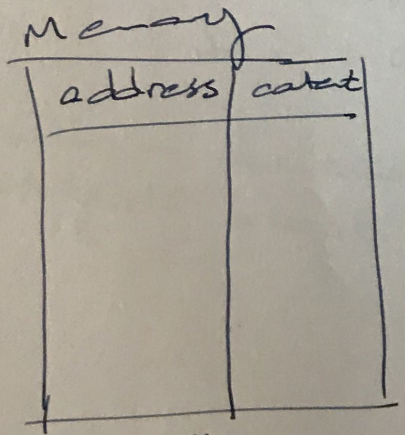
~~int a;~~

→ Registers

→ Memory

→ Write code





Arm architecture "requires" performing all operations on registers.

Arm Assembly Emulators

→ VisUAL

mov → Assembly instruction.

mov R0, #0x12

Assign the hexadecimal value 0x12 to the register R0.

mov R0, R1
"copy" the content of register R1 to R0.
mov only works for register contents.

mov destination, source
double operand instruction.

ADD R1, R0, #0x4
sources
destination
R1 = R0 + 0x4

LDR → Load memory content to Register
STR → Store Register content to Memory

C Language Exaples

Listing 3.7	Painter usage exaple
Listing 3.8	Array usage exaple
Listing 3.9	Structure usage exaple
Listing 3.10	Painter to structure
Listing 3.11	Define statement and castant declaration
Listing 3.12	Casting in arithmetic operations
Listing 3.13	Reaching a specific menory address
Listing 3.14	Usage of the "math" header

Visual Assembly Codes

asm-exaple1 → basic operations

asm-exaple 2 → reaching memory

asm-exaple 3 → reaching memory

asm-exaple3-1 → memory operations

asm-exaple 3-2 → memory operations + arithmetic and logic operations

asm-exaple 4 → arithmetic and logic ops.

asm-exaple 5 → conditional statements, loops

asm-exaple 6 → subroutine

asm-exaple 7 → bit banding (Extra)