COMP1521 Tutorial 05

MIPS Variables

Give MIPS directives to represent the following variables

- int v0;
- int v1 = 42;
- char v2;
- char v3 = 'a';
- double v4;
- int v5[20];
- int v6[10][5];
- struct { int x; int y; } v7;
- struct { int x; int y; } v8[4];
- struct { int x; int y; } *v9[4];

Stack Frames in MIPS

- Stack frames store local variables used in function instances
  - *Frame Pointer ($fp)* points to the top of the current functions stack frame
  - *Stack Pointer ($sp)* points to the next available memory address in stack
- Created upon function call, released once function returns
- Follows stack ADT: First-in, Last-out

- Allows easy organisation and structuring of memory in the procedural programming paradigm

Stack Frames in MIPS

```
#  set up stack frame
   sw    $fp, -4($sp)       # push $fp onto stack
   la    $fp, -4($sp)       # set up $fp for this function
   sw    $ra, -4($fp)       # save return address
   sw    $s0, -8($fp)       # save $s0 to use as ... int n;
   sw    $t1, -12($fp)      # save $t1 to stack
   addi  $sp, $sp, -16      # reset $sp to last pushed item
```

```
# clean up stack frame
  lw    $t1, -12($fp)
  lw    $s0, -8($fp)       # restore $s0 value
  lw    $ra, -4($fp)       # restore $ra for return
  la    $sp, 4($fp)        # restore $sp (remove stack frame)
  lw    $fp, ($fp)          # restore $fp (remove stack frame)
```

C to MIPS

```
 1. int max(int a[], int length)
 2. {
 3.    if (length == 1)
 4.       return a[0];
 5.    else {
 6.       // find max value in rest of array
 7.       int max_so_far = max(&a[1], length-1);
 8.       // check if it's bigger than the first element
 9.       return (a[0] > max_so_far) ? a[0] : max_so_far;
10.    }
11. }
```

2D Arrays in MIPS

```
nrows:  .word 6
ncols:  .word 12
flag:   .byte '#', '#', '#', '#', '#', '.', '.', '#', '#', '#', '#', '#'
        .byte '#', '#', '#', '#', '#', '.', '.', '#', '#', '#', '#', '#'
        .byte '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'
        .byte '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.', '.'
        .byte '#', '#', '#', '#', '#', '.', '.', '#', '#', '#', '#', '#'
        .byte '#', '#', '#', '#', '#', '.', '.', '#', '#', '#', '#', '#'
```

Multiplying a matrix in MIPS

```
1. void change(int nrows, int ncols, int M[nrows][ncols], int factor)
2. {
3.    for (int row = 0; row < nrows; row++) {
4.       for (int col = 0; col < ncols; col++) {
5.          M[row][col] = factor * M[row][col];
6.       }
7.    }
8. }
9.
```

```
   li   $a0, 3
   li   $a1, 4
   la   $a2, M
   li   $a3, 2
   jal  change
   ...
```

```
# where M is defined as
M:   .word 1, 2, 3, 4
     .word 3, 4, 5, 6
     .word 5, 6, 7, 8
```

3D Array in MIPS