

Week 10 Review

Question 1

- What are the offsets of the fields?

```
typedef struct {  
    char a[5]; // this is an array of 5 chars  
    short b;  
    int c;  
} Foo;
```

Question 1

- What are the offsets of the fields?

```
typedef struct {  
    char a[5]; // this is an array of 5 chars  
    short b;  
    int c;  
} Foo;
```

Offset	What?
0	
1	
2	
...	

Question 1

- What are the offsets of the fields?

```
typedef struct {  
    char a[5]; // this is an array of 5 chars  
    short b;  
    int c;  
} Foo;
```

- **a → 0**
 - First field is always at offset 0...
- **b → 6**
 - Previous field ends at 5, so pad to half-word alignment
- **c → 8**
 - Previous field ends at 8, which is word-aligned.

Offset	What?
0	a[0]
1	a[1]
2	a[2]
3	a[3]
4	a[4]
5	padding
6	b
7	b
8	c
9	c
10	c
11	c

Question 2

Final Exam, Winter 2012:

3. In the space below, write a short assembly language program that is a translation of the program on the right. You can assume that `i` has been placed on the top of the stack, and that the return value should be placed on the stack as well before returning to the calling program. Make sure that you comment your code so that we understand what you're doing. (10 marks)

```
int sign (int i) {
    if (i > 0)
        return 1;
    else if (i < 0)
        return -1;
    else
        return 0;
}
```

- How would you convert this to assembly language?

Reference Information

ALU arithmetic input table:

Select	Input	Operation
S ₁ S ₀	Y	G _A =0 G _A =1
0 0	All 0s	G=A
0 1	B	G=A+B
1 0	B	G=A-B
1 1	All 1s	G=A

Register table:

- Register values: Processor role**
- Register 0 (Zero): value 0.
 - Register 1 (Sat): reserved for the assembler.
 - Registers 2-3 (S₀, S₁): return values
 - Registers 4-7 (S₀, S₁): function arguments
 - Registers 8-15, 24-25 (S₀-S₁): temporaries
 - Registers 16-23 (S₀-S₁): saved temporaries
 - Registers 28-31 (S₀, S₁, S₂, S₃)

Instruction table:

Instruction	Op/Func	Syntax
<code>add</code>	100000	\$d, \$s, \$t
<code>addu</code>	100001	\$d, \$s, \$t
<code>addi</code>	001000	\$t, \$s, i
<code>addiu</code>	001001	\$t, \$s, i
<code>and</code>	011000	\$s, \$t
<code>andi</code>	011001	\$s, \$t
<code>andl</code>	011010	\$s, \$t
<code>andhi</code>	011011	\$s, \$t
<code>andb</code>	011012	\$s, \$t
<code>andbl</code>	011013	\$s, \$t
<code>andbr</code>	011014	\$s, \$t
<code>andbrl</code>	011015	\$s, \$t
<code>andbrh</code>	011016	\$s, \$t
<code>andbrl</code>	011017	\$s, \$t
<code>andbrh</code>	011018	\$s, \$t
<code>andbrl</code>	011019	\$s, \$t
<code>andbrh</code>	011020	\$s, \$t
<code>andbrl</code>	011021	\$s, \$t
<code>andbrh</code>	011022	\$s, \$t
<code>andbrl</code>	011023	\$s, \$t
<code>andbrh</code>	011024	\$s, \$t
<code>andbrl</code>	011025	\$s, \$t
<code>andbrh</code>	011026	\$s, \$t
<code>andbrl</code>	011027	\$s, \$t
<code>andbrh</code>	011028	\$s, \$t
<code>andbrl</code>	011029	\$s, \$t
<code>andbrh</code>	011030	\$s, \$t
<code>andbrl</code>	011031	\$s, \$t
<code>andbrh</code>	011032	\$s, \$t
<code>andbrl</code>	011033	\$s, \$t
<code>andbrh</code>	011034	\$s, \$t
<code>andbrl</code>	011035	\$s, \$t
<code>andbrh</code>	011036	\$s, \$t
<code>andbrl</code>	011037	\$s, \$t
<code>andbrh</code>	011038	\$s, \$t
<code>andbrl</code>	011039	\$s, \$t
<code>andbrh</code>	011040	\$s, \$t
<code>andbrl</code>	011041	\$s, \$t
<code>andbrh</code>	011042	\$s, \$t
<code>andbrl</code>	011043	\$s, \$t
<code>andbrh</code>	011044	\$s, \$t
<code>andbrl</code>	011045	\$s, \$t
<code>andbrh</code>	011046	\$s, \$t
<code>andbrl</code>	011047	\$s, \$t
<code>andbrh</code>	011048	\$s, \$t
<code>andbrl</code>	011049	\$s, \$t
<code>andbrh</code>	011050	\$s, \$t
<code>andbrl</code>	011051	\$s, \$t
<code>andbrh</code>	011052	\$s, \$t
<code>andbrl</code>	011053	\$s, \$t
<code>andbrh</code>	011054	\$s, \$t
<code>andbrl</code>	011055	\$s, \$t
<code>andbrh</code>	011056	\$s, \$t
<code>andbrl</code>	011057	\$s, \$t
<code>andbrh</code>	011058	\$s, \$t
<code>andbrl</code>	011059	\$s, \$t
<code>andbrh</code>	011060	\$s, \$t
<code>andbrl</code>	011061	\$s, \$t
<code>andbrh</code>	011062	\$s, \$t
<code>andbrl</code>	011063	\$s, \$t
<code>andbrh</code>	011064	\$s, \$t
<code>andbrl</code>	011065	\$s, \$t
<code>andbrh</code>	011066	\$s, \$t
<code>andbrl</code>	011067	\$s, \$t
<code>andbrh</code>	011068	\$s, \$t
<code>andbrl</code>	011069	\$s, \$t
<code>andbrh</code>	011070	\$s, \$t
<code>andbrl</code>	011071	\$s, \$t
<code>andbrh</code>	011072	\$s, \$t
<code>andbrl</code>	011073	\$s, \$t
<code>andbrh</code>	011074	\$s, \$t
<code>andbrl</code>	011075	\$s, \$t
<code>andbrh</code>	011076	\$s, \$t
<code>andbrl</code>	011077	\$s, \$t
<code>andbrh</code>	011078	\$s, \$t
<code>andbrl</code>	011079	\$s, \$t
<code>andbrh</code>	011080	\$s, \$t
<code>andbrl</code>	011081	\$s, \$t
<code>andbrh</code>	011082	\$s, \$t
<code>andbrl</code>	011083	\$s, \$t
<code>andbrh</code>	011084	\$s, \$t
<code>andbrl</code>	011085	\$s, \$t
<code>andbrh</code>	011086	\$s, \$t
<code>andbrl</code>	011087	\$s, \$t
<code>andbrh</code>	011088	\$s, \$t
<code>andbrl</code>	011089	\$s, \$t
<code>andbrh</code>	011090	\$s, \$t
<code>andbrl</code>	011091	\$s, \$t
<code>andbrh</code>	011092	\$s, \$t
<code>andbrl</code>	011093	\$s, \$t
<code>andbrh</code>	011094	\$s, \$t
<code>andbrl</code>	011095	\$s, \$t
<code>andbrh</code>	011096	\$s, \$t
<code>andbrl</code>	011097	\$s, \$t
<code>andbrh</code>	011098	\$s, \$t
<code>andbrl</code>	011099	\$s, \$t
<code>andbrh</code>	011100	\$s, \$t
<code>andbrl</code>	011101	\$s, \$t
<code>andbrh</code>	011102	\$s, \$t
<code>andbrl</code>	011103	\$s, \$t
<code>andbrh</code>	011104	\$s, \$t
<code>andbrl</code>	011105	\$s, \$t
<code>andbrh</code>	011106	\$s, \$t
<code>andbrl</code>	011107	\$s, \$t
<code>andbrh</code>	011108	\$s, \$t
<code>andbrl</code>	011109	\$s, \$t
<code>andbrh</code>	011110	\$s, \$t
<code>andbrl</code>	011111	\$s, \$t
<code>andbrh</code>	011112	\$s, \$t
<code>andbrl</code>	011113	\$s, \$t
<code>andbrh</code>	011114	\$s, \$t
<code>andbrl</code>	011115	\$s, \$t
<code>andbrh</code>	011116	\$s, \$t
<code>andbrl</code>	011117	\$s, \$t
<code>andbrh</code>	011118	\$s, \$t
<code>andbrl</code>	011119	\$s, \$t
<code>andbrh</code>	011120	\$s, \$t
<code>andbrl</code>	011121	\$s, \$t
<code>andbrh</code>	011122	\$s, \$t
<code>andbrl</code>	011123	\$s, \$t
<code>andbrh</code>	011124	\$s, \$t
<code>andbrl</code>	011125	\$s, \$t
<code>andbrh</code>	011126	\$s, \$t
<code>andbrl</code>	011127	\$s, \$t
<code>andbrh</code>	011128	\$s, \$t
<code>andbrl</code>	011129	\$s, \$t
<code>andbrh</code>	011130	\$s, \$t
<code>andbrl</code>	011131	\$s, \$t
<code>andbrh</code>	011132	\$s, \$t
<code>andbrl</code>	011133	\$s, \$t
<code>andbrh</code>	011134	\$s, \$t
<code>andbrl</code>	011135	\$s, \$t
<code>andbrh</code>	011136	\$s, \$t
<code>andbrl</code>	011137	\$s, \$t
<code>andbrh</code>	011138	\$s, \$t
<code>andbrl</code>	011139	\$s, \$t
<code>andbrh</code>	011140	\$s, \$t
<code>andbrl</code>	011141	\$s, \$t
<code>andbrh</code>	011142	\$s, \$t
<code>andbrl</code>	011143	\$s, \$t
<code>andbrh</code>	011144	\$s, \$t
<code>andbrl</code>	011145	\$s, \$t
<code>andbrh</code>	011146	\$s, \$t
<code>andbrl</code>	011147	\$s, \$t
<code>andbrh</code>	011148	\$s, \$t
<code>andbrl</code>	011149	\$s, \$t
<code>andbrh</code>	011150	\$s, \$t
<code>andbrl</code>	011151	\$s, \$t
<code>andbrh</code>	011152	\$s, \$t
<code>andbrl</code>	011153	\$s, \$t
<code>andbrh</code>	011154	\$s, \$t
<code>andbrl</code>	011155	\$s, \$t
<code>andbrh</code>	011156	\$s, \$t
<code>andbrl</code>	011157	\$s, \$t
<code>andbrh</code>	011158	\$s, \$t
<code>andbrl</code>	011159	\$s, \$t
<code>andbrh</code>	011160	\$s, \$t
<code>andbrl</code>	011161	\$s, \$t
<code>andbrh</code>	011162	\$s, \$t
<code>andbrl</code>	011163	\$s, \$t
<code>andbrh</code>	011164	\$s, \$t
<code>andbrl</code>	011165	\$s, \$t
<code>andbrh</code>	011166	\$s, \$t
<code>andbrl</code>	011167	\$s, \$t
<code>andbrh</code>	011168	\$s, \$t
<code>andbrl</code>	011169	\$s, \$t
<code>andbrh</code>	011170	\$s, \$t
<code>andbrl</code>	011171	\$s, \$t
<code>andbrh</code>	011172	\$s, \$t
<code>andbrl</code>	011173	\$s, \$t
<code>andbrh</code>	011174	\$s, \$t
<code>andbrl</code>	011175	\$s, \$t
<code>andbrh</code>	011176	\$s, \$t
<code>andbrl</code>	011177	\$s, \$t
<code>andbrh</code>	011178	\$s, \$t
<code>andbrl</code>	011179	\$s, \$t
<code>andbrh</code>	011180	\$s, \$t
<code>andbrl</code>	011181	\$s, \$t
<code>andbrh</code>	011182	\$s, \$t
<code>andbrl</code>	011183	\$s, \$t
<code>andbrh</code>	011184	\$s, \$t
<code>andbrl</code>	011185	\$s, \$t
<code>andbrh</code>	011186	\$s, \$t
<code>andbrl</code>	011187	\$s, \$t
<code>andbrh</code>	011188	\$s, \$t
<code>andbrl</code>	011189	\$s, \$t
<code>andbrh</code>	011190	\$s, \$t
<code>andbrl</code>	011191	\$s, \$t
<code>andbrh</code>	011192	\$s, \$t
<code>andbrl</code>	011193	\$s, \$t
<code>andbrh</code>	011194	\$s, \$t
<code>andbrl</code>	011195	\$s, \$t
<code>andbrh</code>	011196	\$s, \$t
<code>andbrl</code>	011197	\$s, \$t
<code>andbrh</code>	011198	\$s, \$t
<code>andbrl</code>	011199	\$s, \$t
<code>andbrh</code>	011200	\$s, \$t
<code>andbrl</code>	011201	\$s, \$t
<code>andbrh</code>	011202	\$s, \$t
<code>andbrl</code>	011203	\$s, \$t
<code>andbrh</code>	011204	\$s, \$t
<code>andbrl</code>	011205	\$s, \$t
<code>andbrh</code>	011206	\$s, \$t
<code>andbrl</code>	011207	\$s, \$t
<code>andbrh</code>	011208	\$s, \$t
<code>andbrl</code>	011209	\$s, \$t
<code>andbrh</code>	011210	\$s, \$t
<code>andbrl</code>	011211	\$s, \$t
<code>andbrh</code>	011212	\$s, \$t
<code>andbrl</code>	011213	\$s, \$t
<code>andbrh</code>	011214	\$s, \$t
<code>andbrl</code>	011215	\$s, \$t
<code>andbrh</code>	011216	\$s, \$t
<code>andbrl</code>	011217	\$s, \$t
<code>andbrh</code>	011218	\$s, \$t
<code>andbrl</code>	011219	\$s, \$t
<code>andbrh</code>	011220	\$s, \$t
<code>andbrl</code>	011221	\$s, \$t
<code>andbrh</code>	011222	\$s, \$t
<code>andbrl</code>	011223	\$s, \$t
<code>andbrh</code>	011224	\$s, \$t
<code>andbrl</code>	011225	\$s, \$t
<code>andbrh</code>	011226	\$s, \$t
<code>andbrl</code>	011227	\$s, \$t
<code>andbrh</code>	011228	\$s, \$t
<code>andbrl</code>	011229	\$s, \$t
<code>andbrh</code>	011230	\$s, \$t
<code>andbrl</code>	011231	\$s, \$t
<code>andbrh</code>	011232	\$s, \$t
<code>andbrl</code>	011233	\$s, \$t
<code>andbrh</code>	011234	\$s, \$t
<code>andbrl</code>	011235	\$s, \$t
<code>andbrh</code>	011236	\$s, \$t
<code>andbrl</code>	011237	\$s, \$t
<code>andbrh</code>	011238	\$s, \$t
<code>andbrl</code>	011239	\$s, \$t
<code>andbrh</code>	011240	\$s, \$t
<code>andbrl</code>	011241	\$s, \$t
<code>andbrh</code>	011242	\$s, \$t
<code>andbrl</code>	011243	\$s, \$t
<code>andbrh</code>	011244	\$s, \$t
<code>andbrl</code>	011245	\$s, \$t
<code>andbrh</code>	011246	\$s, \$t
<code>andbrl</code>	011247	\$s, \$t
<code>andbrh</code>	011248	\$s, \$t
<code>andbrl</code>	011249	\$s, \$t
<code>andbrh</code>	011250	\$s, \$t
<code>andbrl</code>	011251	\$s, \$t
<code>andbrh</code>	011252	\$s, \$t
<code>andbrl</code>	011253	\$s, \$t
<code>andbrh</code>	011254	\$s, \$t
<code>andbrl</code>	011255	\$s, \$t
<code>andbrh</code>	011256	\$s, \$t
<code>andbrl</code>	011257	\$s, \$t
<code>andbrh</code>	011258	\$s, \$t
<code>andbrl</code>	011259	\$s, \$t
<code>andbrh</code>	011260	\$s, \$t
<code>andbrl</code>	011261	\$s, \$t
<code>andbrh</code>	011262	\$s, \$t
<code>andbrl</code>	011263	\$s, \$t
<code>andbrh</code>	011264	\$s, \$t
<code>andbrl</code>	011265	\$s, \$t
<code>andbrh</code>	011266	\$s, \$t
<code>andbrl</code>	011267	\$s, \$t
<code>andbrh</code>	011268	\$s, \$t
<code>andbrl</code>	011269	\$s, \$t
<code>andbrh</code>	011270	\$s, \$t
<code>andbrl</code>	011271	\$s, \$t
<code>andbrh</code>	011272	\$s, \$t
<code>andbrl</code>	011273	\$s, \$t
<code>andbrh</code>	011274	\$s, \$t
<code>andbrl</code>	011275	\$s, \$t
<code>andbrh</code>	011276	\$s, \$t
<code>andbrl</code>	011277	\$s, \$t
<code>andbrh</code>	011278	\$s, \$t
<code>andbrl</code>	011279	\$s, \$t
<code>andbrh</code>	011280	\$s, \$t
<code>andbrl</code>	011281	\$s, \$t
<code>andbrh</code>	011282	\$s, \$t
<code>andbrl</code>	011283	\$s, \$t
<code>andbrh</code>	011284	\$s, \$t
<code>andbrl</code>	011285	\$s, \$t
<code>andbrh</code>	011286	\$s, \$t
<code>andbrl</code>	011287	\$s, \$t
<code>andbrh</code>	011288	\$s, \$t
<code>andbrl</code>	011289	\$s, \$t
<code>andbrh</code>	011290	\$s, \$t
<code>andbrl</code>	011291	\$s, \$t
<code>andbrh</code>	011292	\$s, \$t
<code>andbrl</code>	011293	\$s, \$t
<code>andbrh</code>	011294	\$s, \$t
<code>andbrl</code>	011295	\$s, \$t
<code>andbrh</code>	011296	\$s, \$t
<code>andbrl</code>	011297	\$s, \$t
<code>andbrh</code>	011298	\$s, \$t
<code>andbrl</code>	011299	\$s, \$t
<code>andbrh</code>	011300	\$s, \$t
<code>andbrl</code>	011301	\$s,

Code for sign(i)

```
# registers: $t0 = i, $t1 = return value
sign:  lw $t0, 0($sp)          # pop i off the stack
      addi $sp, $sp, 4

      # handle case of i > 0
if_gt: blez $t0, if_lt
      addi $t1, $zero, 1
      j end

      # handle case of i < 0
if_lt: beq $t0, $zero, eq
      addi $t1, $zero, -1
      j end

      # handle case of i == 0
eq:    add $t1, $zero, $zero

end:   addi $sp, $sp, -4      # push return value
      sw $t1, 0($sp)       # return to caller
      jr $ra
```

Question 3: Remember this?

- What does the following function do?:

```
myfunc: lw $t0, 0($sp)
      addi $sp, $sp, 4
      addi $t1, $zero, 2
      div $t0, $t1
      mfhi $t0
      beq $t0, $zero, LABEL1
      add $t2, $zero, $zero
      j LABEL2

LABEL1: addi $t2, $zero, 1
LABEL2: addi $sp, $sp, -4
      sw $t2, 0($sp)
      jr $ra
```

Question 3

- We divided `$t0` by 2
 - `div` puts remainder in `HI`
- If remainder is 0 \rightarrow return 1
- if remainder is not 0 \rightarrow return 0
- This is a function that returns 1 if a number is even or 0 if it is odd

Question 4

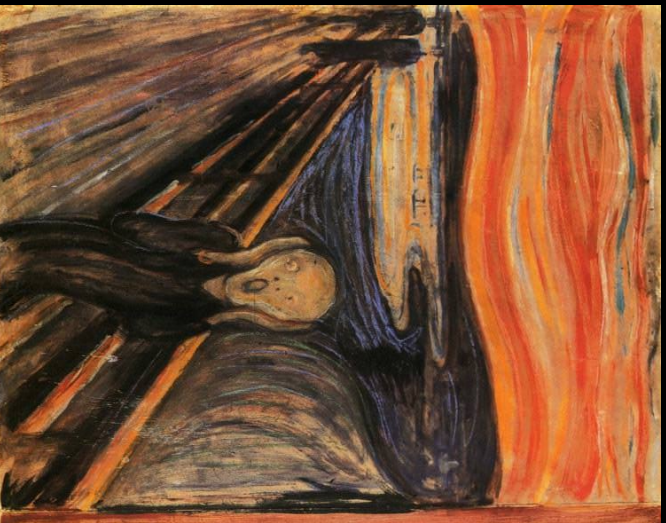
- Write a function `numpLus(arr, n)` that returns the number of elements larger than zero in `arr` minus the number of negative elements.
- Weird restrictions:
 - Not allowed to use loops.
 - Only allowed one branch, it must be `bne` or `beq`.
 - Not allowed any other branches...
 - ...but can use `sign` function.
- How would you do it in C?

Question 4

- Since $\text{sign}(i)$ is -1,0,1, we can compute the sum of $\text{sign}(\text{arr}[i])$ for all i .
- Use recursion instead of loops.

```
int numPlus(int arr[], int n)
{
    if (n == 0) // here we use the branch
        return 0;
    return sign(arr[0]) + numPlus(arr+1, n-1);
}
```

Question 4



Question 4

- **Don't panic.**
- Remember your training!
 - A recursive function is just another function.
 - Save \$ra and any other registers.
- Use "assembly pseudocode" then convert to assembly



Question 4

- Strategy:
 - Pop arguments into \$to, \$t1
 - Deal with base case
 - Save \$ra
 - \$t2 = sign(arr[o])
 - \$t3 = numplus(...)
 - Compute \$t2+\$t3
 - Restore \$ra
 - Return result

Which registers
must be saved in
each case?

Question 4

- Pop arr, n
- Check for base case $n == 0$:
 - Return 0 if n is zero
- Save \$ra
- Push arr, n to save them
- Get arr[0]
- Call sign(arr[0])
- Pop result into \$t2
- Pop arr, n to restore them
- Push \$t2 to save it (it contains the result of sign(...))
- Call numplus(arr+4, n-1) ← **note arr+4 (in C it would be +1)**
- Pop return value into \$t3
- Pop \$t2 to restore it
- Restore \$ra
- Return \$t2 + \$t3