Assembling vm520 Programs

cs520

Dept. of Computer Science
Univ. of New Hampshire

The Assembly Process

translate human-readable program
to machine-readable program

two passes

1. Determine addresses for all labels.
2. Generate the machine code. → store in symbol table
   → a.k.a. object code

Why two passes?

can be use of label before
it is defined
i.e. forward reference

```
#
# This vm520 program simply sums together a list of numbers.
#
# The label "sum" is exported to allow the main program invoking
# the virtual machine to retrieve the answer.
#
# The labels "top" and "done" are exported to allow tests of the disassembler.
#

export sum
export top
export done
```

Pass 1 → Address

Pass 2 → Contents

	Address	Contents	
`jmp skipData`	0	00007014	
`sum:`			
`  word 0`	1	00000000	
`len:`			
`  word 5`	2	00000005	
`vector:`			
`  word 1`	3	00000001	
`  word 2`	4	00000002	
`  word 3`	5	00000003	
`  word 4`	6	00000004	
`  word 5`	7	00000005	
`skipData:`			
`  ldimm  r0, 0`	8	00000003	
`  load   r1, len`	9	FFFF8101	
`  ldaddr r2, vector`	10	FFFF8204	
`  ldimm  r3, 0`	11	00000303	
`  ldimm  r4, 1`	12	00001403	
`top:`			
`  beq    r0, r1, done`	13	00051013	
`  ldind  r5, 0(r2)`	14	00002505	
`  addi   r3, r5`	15	00005306	
`  addi   r2, r4`	16	00004208	
`  addi   r0, r4`	17	00004008	
`  jmp    top`	18	FFFFA014	
`done:`			
`  store  r3, sum`	19	FFFED302	
`  halt`	20	00000060	

Symbol T.S.

(sum, ex 1) (skipData, 8)
(len, 2) (top, ex 13)
(vector, 3) (done, ex 19)

Object F.G

contains machine code and associated symbol information

symbol information:

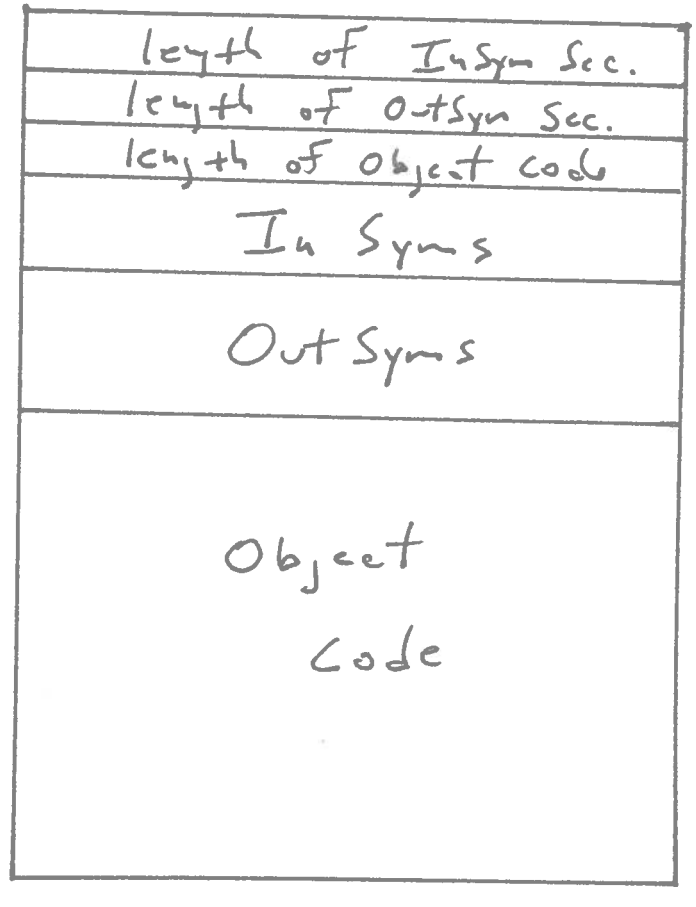
insymbols: labels defined in the machine code and exported to the linker

outsymbols: labels referenced in the machine code but not defined in the machine code

linker – combines object F.G's together matches outsymbols in one F.G to insymbols in other F.G

Vm520 Object File

series of 32-bit words in Little-Endian Format



Symbols in vm520 Object File

each symbol stored in 5 words:

words 1–4: name

word 5: address** → InSym: definition address

→ OutSym: address of reference*

↳ a symbol can appear more than once in OutSym section

example: (done, 19_{10})

```
656E6F64  ← 'e' 'n' 'o' 'd'
00000000
00000000   } null fill
00 0000 00
00 00 00 13  → 13₁₆ = 19₁₀
```

* address of the instruction referencing the symbol.

** first word in object code is assumed to be address 0.

```
0000000 0000000f 00000000 00000015 656e6f64
0000020 00000000 00000000 00000000 00000013
0000040 00706f74 00000000 00000000 00000000
0000060 0000000d 006d7573 00000000 00000000
0000100 00000000 00000001 00007014 00000000
0000120 00000005 00000001 00000002 00000003
0000140 00000004 00000005 00000003 ffff8101
0000160 ffff8204 00000303 00001403 00051013
0000200 00002505 0000530b 0000420b 0000400b
0000220 ffffa014 fffed302 00000000
0000234
```

done

sum

od -tx4 sumVector.obj