CS 537: Intro to Operating Systems (Fall 2017)

Worksheet 4 - Memory Virtualization

Due: Oct 4th 2017 (Wed) in-class OR email to Simmi before 11:59 pm

1. **Simple Paging**

   Assume a system with a simple linear page table with the following parameters:

   - Size of virtual address space = 512 bytes
   - Size of physical memory = 1024 bytes
   - Page size = 64 bytes
   - Size of page table entry (PTE) = 1 byte
   - Value of page table base register (PTBR) = 0x1d0

   The left most bit (MSB) in the PTE is the valid bit and it determines if the virtual page is valid or not (1 = valid page; 0 = invalid page). The 4 right most bits (LSBs) in a PTE contains the physical frame number (PFN). You may assume that all the other bits are unused. The contents of the page table are shown below.

Entry 0	0x85
Entry 1	0x8a
Entry 2	0x7c
Entry 3	0x57
Entry 4	0x08
Entry 5	0x89
Entry 6	0x23
Entry 7	0x8d

   The instruction below loads a single byte from virtual address 372 into the register %eax. This instruction resides at virtual address 24 within the address space of the process

   24: mov 372, %eax

   In the diagram of **physical memory**, where each number (in hex) represents a physical address, shown on the next page:

   (a) Put a **BOX** around the **page table** and label it.

   (b) Put a **BOX** around each **valid virtual page** and label them.

   (c) **CIRCLE** the memory addresses that get referenced during the execution of the instruction, including both the instruction fetch and data access. Remember to include the memory references to the PTEs in the page table too!

   (d) **LABEL** the memory address (that you circled) with a **NUMBER** that indicates the **ORDER** in which various physical addresses get referenced.

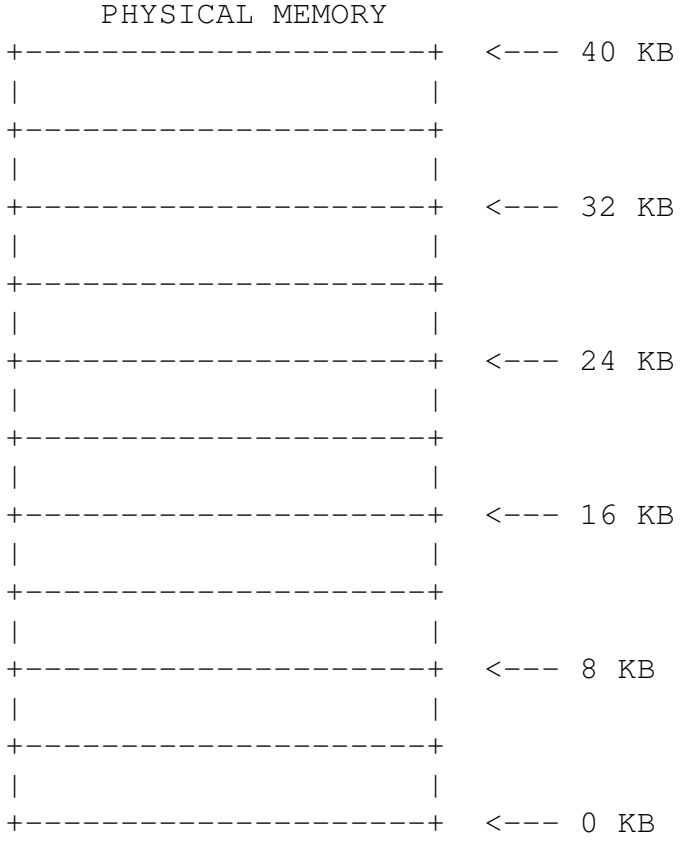
000	001	002	003	004	005	006	007	008	009	00a	00b	00c	00d	00e	00f
010	011	012	013	014	015	016	017	018	019	01a	01b	01c	01d	01e	01f
020	021	022	023	024	025	026	027	028	029	02a	02b	02c	02d	02e	02f
030	031	032	033	034	035	036	037	038	039	03a	03b	03c	03d	03e	03f
040	041	042	043	044	045	046	047	048	049	04a	04b	04c	04d	04e	04f
050	051	052	053	054	055	056	057	058	059	05a	05b	05c	05d	05e	05f
060	061	062	063	064	065	066	067	068	069	06a	06b	06c	06d	06e	06f
070	071	072	073	074	075	076	077	078	079	07a	07b	07c	07d	07e	07f
080	081	082	083	084	085	086	087	088	089	08a	08b	08c	08d	08e	08f
090	091	092	093	094	095	096	097	098	099	09a	09b	09c	09d	09e	09f
0a0	0a1	0a2	0a3	0a4	0a5	0a6	0a7	0a8	0a9	0aa	0ab	0ac	0ad	0ae	0af
0b0	0b1	0b2	0b3	0b4	0b5	0b6	0b7	0b8	0b9	0ba	0bb	0bc	0bd	0be	0bf
0c0	0c1	0c2	0c3	0c4	0c5	0c6	0c7	0c8	0c9	0ca	0cb	0cc	0cd	0ce	0cf
0d0	0d1	0d2	0d3	0d4	0d5	0d6	0d7	0d8	0d9	0da	0db	0dc	0dd	0de	0df
0e0	0e1	0e2	0e3	0e4	0e5	0e6	0e7	0e8	0e9	0ea	0eb	0ec	0ed	0ee	0ef
0f0	0f1	0f2	0f3	0f4	0f5	0f6	0f7	0f8	0f9	0fa	0fb	0fc	0fd	0fe	0ff
100	101	102	103	104	105	106	107	108	109	10a	10b	10c	10d	10e	10f
110	111	112	113	114	115	116	117	118	119	11a	11b	11c	11d	11e	11f
120	121	122	123	124	125	126	127	128	129	12a	12b	12c	12d	12e	12f
130	131	132	133	134	135	136	137	138	139	13a	13b	13c	13d	13e	13f
140	141	142	143	144	145	146	147	148	149	14a	14b	14c	14d	14e	14f
150	151	152	153	154	155	156	157	158	159	15a	15b	15c	15d	15e	15f
160	161	162	163	164	165	166	167	168	169	16a	16b	16c	16d	16e	16f
170	171	172	173	174	175	176	177	178	179	17a	17b	17c	17d	17e	17f
180	181	182	183	184	185	186	187	188	189	18a	18b	18c	18d	18e	18f
190	191	192	193	194	195	196	197	198	199	19a	19b	19c	19d	19e	19f
1a0	1a1	1a2	1a3	1a4	1a5	1a6	1a7	1a8	1a9	1aa	1ab	1ac	1ad	1ae	1af
1b0	1b1	1b2	1b3	1b4	1b5	1b6	1b7	1b8	1b9	1ba	1bb	1bc	1bd	1be	1bf
1c0	1c1	1c2	1c3	1c4	1c5	1c6	1c7	1c8	1c9	1ca	1cb	1cc	1cd	1ce	1cf
1d0	1d1	1d2	1d3	1d4	1d5	1d6	1d7	1d8	1d9	1da	1db	1dc	1dd	1de	1df
1e0	1e1	1e2	1e3	1e4	1e5	1e6	1e7	1e8	1e9	1ea	1eb	1ec	1ed	1ee	1ef
1f0	1f1	1f2	1f3	1f4	1f5	1f6	1f7	1f8	1f9	1fa	1fb	1fc	1fd	1fe	1ff
200	201	202	203	204	205	206	207	208	209	20a	20b	20c	20d	20e	20f
210	211	212	213	214	215	216	217	218	219	21a	21b	21c	21d	21e	21f
220	221	222	223	224	225	226	227	228	229	22a	22b	22c	22d	22e	22f
230	231	232	233	234	235	236	237	238	239	23a	23b	23c	23d	23e	23f
240	241	242	243	244	245	246	247	248	249	24a	24b	24c	24d	24e	24f
250	251	252	253	254	255	256	257	258	259	25a	25b	25c	25d	25e	25f
260	261	262	263	264	265	266	267	268	269	26a	26b	26c	26d	26e	26f
270	271	272	273	274	275	276	277	278	279	27a	27b	27c	27d	27e	27f
280	281	282	283	284	285	286	287	288	289	28a	28b	28c	28d	28e	28f
290	291	292	293	294	295	296	297	298	299	29a	29b	29c	29d	29e	29f
2a0	2a1	2a2	2a3	2a4	2a5	2a6	2a7	2a8	2a9	2aa	2ab	2ac	2ad	2ae	2af
2b0	2b1	2b2	2b3	2b4	2b5	2b6	2b7	2b8	2b9	2ba	2bb	2bc	2bd	2be	2bf
2c0	2c1	2c2	2c3	2c4	2c5	2c6	2c7	2c8	2c9	2ca	2cb	2cc	2cd	2ce	2cf
2d0	2d1	2d2	2d3	2d4	2d5	2d6	2d7	2d8	2d9	2da	2db	2dc	2dd	2de	2df
2e0	2e1	2e2	2e3	2e4	2e5	2e6	2e7	2e8	2e9	2ea	2eb	2ec	2ed	2ee	2ef
2f0	2f1	2f2	2f3	2f4	2f5	2f6	2f7	2f8	2f9	2fa	2fb	2fc	2fd	2fe	2ff
300	301	302	303	304	305	306	307	308	309	30a	30b	30c	30d	30e	30f
310	311	312	313	314	315	316	317	318	319	31a	31b	31c	31d	31e	31f
320	321	322	323	324	325	326	327	328	329	32a	32b	32c	32d	32e	32f
330	331	332	333	334	335	336	337	338	339	33a	33b	33c	33d	33e	33f
340	341	342	343	344	345	346	347	348	349	34a	34b	34c	34d	34e	34f
350	351	352	353	354	355	356	357	358	359	35a	35b	35c	35d	35e	35f
360	361	362	363	364	365	366	367	368	369	36a	36b	36c	36d	36e	36f
370	371	372	373	374	375	376	377	378	379	37a	37b	37c	37d	37e	37f
380	381	382	383	384	385	386	387	388	389	38a	38b	38c	38d	38e	38f
390	391	392	393	394	395	396	397	398	399	39a	39b	39c	39d	39e	39f
3a0	3a1	3a2	3a3	3a4	3a5	3a6	3a7	3a8	3a9	3aa	3ab	3ac	3ad	3ae	3af
3b0	3b1	3b2	3b3	3b4	3b5	3b6	3b7	3b8	3b9	3ba	3bb	3bc	3bd	3be	3bf
3c0	3c1	3c2	3c3	3c4	3c5	3c6	3c7	3c8	3c9	3ca	3cb	3cc	3cd	3ce	3cf
3d0	3d1	3d2	3d3	3d4	3d5	3d6	3d7	3d8	3d9	3da	3db	3dc	3dd	3de	3df
3e0	3e1	3e2	3e3	3e4	3e5	3e6	3e7	3e8	3e9	3ea	3eb	3ec	3ed	3ee	3ef
3f0	3f1	3f2	3f3	3f4	3f5	3f6	3f7	3f8	3f9	3fa	3fb	3fc	3fd	3fe	3ff

2. **Segmentation**

Assume a system that uses **segmented virtual memory**. There are **three segments**, namely code, heap, and stack. Note that the code and the heap grow towards high addresses while the stack grows towards low addresses. The size of the **virtual address space is 16KB**, and the size of the **physical address space is 40KB**. The top **three** bits (MSBs) in the virtual address are used to identify the segment. 000 and 001 for code; 010, 011 and 100 for heap; 110 and 111 for stack. The value of the **bounds** register for **code, heap, and stack** are **2KB, 4KB, and 3KB** respectively.

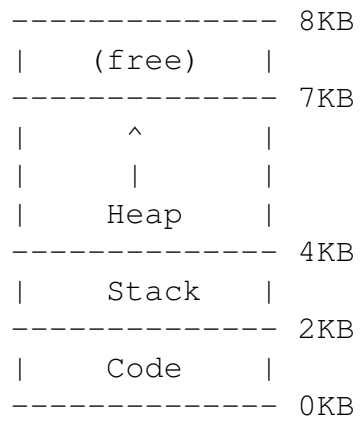
Given the following virtual to physical address translations, sketch the **physical memory** layout below. You should **mark the locations** in physical memory where the three segments, namely code, heap, and stack are placed. You should also label the **start** and **end** addresses (in decimal) of the three segments in physical memory.

Virtual Address	**Physical Address**
6 KB	34 KB
0 KB	8 KB
14 KB	30 KB



3. Base and Bounds

Consider a machine that uses the **base and bounds technique** for memory virtualization and a slightly **different address space** as shown below. Note that the **stack** is of fixed size and it immediately follows the code section. The heap starts at the end of the stack and grows upward. In this configuration, there is only one direction of growth, towards higher regions of the address space. The figure below is NOT drawn to scale.



Parameters:
Size of the virtual address space = 8KB, size of the physical memory = 32KB, base register = 16KB, and bounds register = 7KB. References to any address within the bounds would be considered legal; references above this value are out of bounds and thus the hardware would raise an exception.

a. For each **virtual address**, either write down the **physical address** it translates to OR write down that it is a **segmentation fault** (an out-of-bounds address).

Virtual Address	**Physical Address (in hex or decimal) OR Seg Fault**
0x1000 (decimal: 4096)	
0x1C00 (decimal: 7168)	
0x0000 (decimal: 0)	
0x2000 (decimal: 8192)	
0x0800 (decimal: 2048)	

b. For each physical address, write the corresponding virtual address (in hex or decimal), and the logical segment (code, stack, heap) that the virtual address belongs to.

Physical Address	**Virtual Address**	**Segment**
0x5800 (decimal: 22KB)		
0x4400 (decimal: 17KB)		
0x4C00 (decimal: 19KB)		