```
**********************************************************
   46122 Mon Nov  5 20:40:46 2012
new/usr/src/cmd/sgs/libld/common/machrel.amd.c
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
______unchanged_portion_omitted_
```

```
517 #define REX_B           0x1
518 #define REX_X           0x2
519 #define REX_R           0x4
520 #define REX_W           0x8
521 #define REX_PREFIX      0x40

523 #define REX_RW          (REX_PREFIX | REX_R | REX_W)
524 #define REX_BW          (REX_PREFIX | REX_B | REX_W)
525 #define REX_BRW         (REX_PREFIX | REX_B | REX_R | REX_W)

527 #define REG_ESP         0x4

529 #define INSN_ADDMR      0x03    /* addq mem,reg */
530 #define INSN_ADDIR      0x81    /* addq imm,reg */
531 #define INSN_MOVMR      0x8b    /* movq mem,reg */
532 #define INSN_MOVIR      0xc7    /* movq imm,reg */
533 #define INSN_LEA        0x8d    /* leaq mem,reg */
534 #endif /* ! codereview */

536 static Fixupret
537 tls_fixups(Ofl_desc *ofl, Rel_desc *arsp)
538 {
539         Sym_desc        *sdp = arsp->rel_sym;
540         Word            rtype = arsp->rel_rtype;
541         uchar_t         *offset;

543         offset = (uchar_t *)((uintptr_t)arsp->rel_roffset +
544             (uintptr_t)_elf_getxoff(arsp->rel_isdesc->is_indata) +
545             (uintptr_t)RELAUX_GET_OSDESC(arsp)->os_outdata->d_buf);

547         /*
548          * Note that in certain of the original insn sequences below, the
549          * instructions are not necessarily adjacent
550          */
551 #endif /* ! codereview */
552         if (sdp->sd_ref == REF_DYN_NEED) {
553                 /*
554                  * IE reference model
555                  */
556                 switch (rtype) {
557                 case R_AMD64_TLSGD:
558                         /*
559                          *  GD -> IE
560                          *
561                          * Transition:
562                          *      0x00 .byte 0x66
563                          *      0x01 leaq x@tlsgd(%rip), %rdi
564                          *      0x08 .word 0x6666
565                          *      0x0a rex64
566                          *      0x0b call __tls_get_addr@plt
567                          *      0x10
568                          * To:
569                          *      0x00 movq %fs:0, %rax
570                          *      0x09 addq x@gottpoff(%rip), %rax
571                          *      0x10
572                          */
573                         DBG_CALL(Dbg_reloc_transition(ofl->ofl_lml, M_MACH,
574                             R_AMD64_GOTTPOFF, arsp, ld_reloc_sym_name));
575                         arsp->rel_rtype = R_AMD64_GOTTPOFF;
```



```
576                         arsp->rel_roffset += 8;
577                         arsp->rel_raddend = (Sxword)-4;

579                         /*
580                          * Adjust 'offset' to beginning of instruction
581                          * sequence.
582                          */
583                         offset -= 4;
584                         (void) memcpy(offset, tlsinstr_gd_ie,
585                             sizeof (tlsinstr_gd_ie));
586                         return (FIX_RELOC);

588                 case R_AMD64_PLT32:
589                         /*
590                          * Fixup done via the TLS_GD relocation.
591                          */
592                         DBG_CALL(Dbg_reloc_transition(ofl->ofl_lml, M_MACH,
593                             R_AMD64_NONE, arsp, ld_reloc_sym_name));
594                         return (FIX_DONE);
595                 }
596         }

598         /*
599          * LE reference model
600          */
601         switch (rtype) {
602         case R_AMD64_TLSGD:
603                 /*
604                  * GD -> LE
605                  *
606                  * Transition:
607                  *      0x00 .byte 0x66
608                  *      0x01 leaq x@tlsgd(%rip), %rdi
609                  *      0x08 .word 0x6666
610                  *      0x0a rex64
611                  *      0x0b call __tls_get_addr@plt
612                  *      0x10
613                  * To:
614                  *      0x00 movq %fs:0, %rax
615                  *      0x09 leaq x@tpoff(%rax), %rax
616                  *      0x10
617                  */
618                 DBG_CALL(Dbg_reloc_transition(ofl->ofl_lml, M_MACH,
619                     R_AMD64_TPOFF32, arsp, ld_reloc_sym_name));
620                 arsp->rel_rtype = R_AMD64_TPOFF32;
621                 arsp->rel_roffset += 8;
622                 arsp->rel_raddend = 0;

624                 /*
625                  * Adjust 'offset' to beginning of instruction sequence.
626                  */
627                 offset -= 4;
628                 (void) memcpy(offset, tlsinstr_gd_le, sizeof (tlsinstr_gd_le));
629                 return (FIX_RELOC);

631         case R_AMD64_GOTTPOFF: {
517         case R_AMD64_GOTTPOFF:
632                 /*
633                  * IE -> LE
634                  *
635                  * Transition 1:
636                  *      movq %fs:0, %reg
637                  *      addq x@gottpoff(%rip), %reg
638                  * To:
639                  *      movq %fs:0, %reg
640                  *      leaq x@tpoff(%reg), %reg
```

```
641                  *
642                  * Transition (as a special case):
643                  *      movq %fs:0, %r12/%rsp
644                  *      addq x@gottpoff(%rip), %r12/%rsp
645                  * To:
646                  *      movq %fs:0, %r12/%rsp
647                  *      addq x@tpoff(%rax), %r12/%rsp
648                  *
649                  * Transition 2:
650                  *      movq x@gottpoff(%rip), %reg
651                  *      movq %fs:(%reg), %reg
521                  * Transition:
522                  *      0x00 movq %fs:0, %rax
523                  *      0x09 addq x@gottopoff(%rip), %rax
524                  *      0x10
652                  * To:
653                  *      movq x@tpoff(%reg), %reg
654                  *      movq %fs:(%reg), %reg
526                  *      0x00 movq %fs:0, %rax
527                  *      0x09 leaq x@tpoff(%rax), %rax
528                  *      0x10
655                  */
656                 Conv_inv_buf_t  inv_buf;
657                 uint8_t reg;            /* Register */

659                 offset -= 3;

661                 reg = offset[2] >> 3; /* Encoded dest. reg. operand */

663 #endif /* ! codereview */
664                 DBG_CALL(Dbg_reloc_transition(ofl->ofl_lml, M_MACH,
665                     R_AMD64_TPOFF32, arsp, ld_reloc_sym_name));
666                 arsp->rel_rtype = R_AMD64_TPOFF32;
667                 arsp->rel_raddend = 0;

669                 /*
670                  * This is transition 2, and the special case of form 1 where
671                  * a normal transition would index %rsp or %r12 and need a SIB
672                  * byte in the leaq for which we lack space
673                  */
674                 if ((offset[1] == INSN_MOVMR) ||
675                     ((offset[1] == INSN_ADDMR) && (reg == REG_ESP))) {
676                         /*
677                          * If we needed an extra bit of MOD.reg to refer to
678                          * this register as the dest of the original movq we
679                          * need an extra bit of MOD.rm to refer to it in the
680                          * dest of the replacement movq or addq.
681                          */
682                         if (offset[0] == REX_RW)
683                                 offset[0] = REX_BW;

685                         offset[1] = (offset[1] == INSN_MOVMR) ?
686                             INSN_MOVIR : INSN_ADDIR;
687                         offset[2] = 0xc0 | reg;
530                  * Adjust 'offset' to beginning of instruction sequence.
531                  */
532                 offset -= 12;

689                         return (FIX_RELOC);
690                 } else if (offset[1] == INSN_ADDMR) {
691 #endif /* ! codereview */
692                         /*
693                          * If we needed an extra bit of MOD.reg to refer to
694                          * this register in the dest of the addq we need an
695                          * extra bit of both MOD.reg and MOD.rm to refer to it
696                          * in the source and dest of the leaq
```



```
534                  * Same code sequence used in the GD -> LE transition.
697                          */
698                         if (offset[0] == REX_RW)
699                                 offset[0] = REX_BRW;

701                         offset[1] = INSN_LEA;
702                         offset[2] = 0x80 | (reg << 3) | reg;

536                 (void) memcpy(offset, tlsinstr_gd_le, sizeof (tlsinstr_gd_le));
704                         return (FIX_RELOC);
705                 }
706 #endif /* ! codereview */

708                 ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_REL_BADTLSINS),
709                     conv_reloc_amd64_type(arsp->rel_rtype, 0, &inv_buf),
710                     arsp->rel_isdesc->is_file->ifl_name,
711                     ld_reloc_sym_name(arsp),
712                     arsp->rel_isdesc->is_name,
713                     EC_OFF(arsp->rel_roffset));
714                 return (FIX_ERROR);
715         }
716 #endif /* ! codereview */
717         case R_AMD64_TLSLD:
718                 /*
719                  * LD -> LE
720                  *
721                  * Transition
722                  *      0x00 leaq x1@tlsgd(%rip), %rdi
723                  *      0x07 call __tls_get_addr@plt
724                  *      0x0c
725                  * To:
726                  *      0x00 .byte 0x66
727                  *      0x01 .byte 0x66
728                  *      0x02 .byte 0x66
729                  *      0x03 movq %fs:0, %rax
730                  */
731                 DBG_CALL(Dbg_reloc_transition(ofl->ofl_lml, M_MACH,
732                     R_AMD64_NONE, arsp, ld_reloc_sym_name));
733                 offset -= 3;
734                 (void) memcpy(offset, tlsinstr_ld_le, sizeof (tlsinstr_ld_le));
735                 return (FIX_DONE);

737         case R_AMD64_DTPOFF32:
738                 /*
739                  * LD->LE
740                  *
741                  * Transition:
742                  *      0x00 leaq x1@dtpoff(%rax), %rcx
743                  * To:
744                  *      0x00 leaq x1@tpoff(%rax), %rcx
745                  */
746                 DBG_CALL(Dbg_reloc_transition(ofl->ofl_lml, M_MACH,
747                     R_AMD64_TPOFF32, arsp, ld_reloc_sym_name));
748                 arsp->rel_rtype = R_AMD64_TPOFF32;
749                 arsp->rel_raddend = 0;
750                 return (FIX_RELOC);
751         }

753         return (FIX_RELOC);
754 }

756 static uintptr_t
757 ld_do_activerelocs(Ofl_desc *ofl)
758 {
759         Rel_desc        *arsp;
760         Rel_cachebuf    *rcbp;
```

```
761         Aliste          idx;
762         uintptr_t       return_code = 1;
763         ofl_flag_t      flags = ofl->ofl_flags;

765         if (aplist_nitems(ofl->ofl_actrels.rc_list) != 0)
766                 DBG_CALL(Dbg_reloc_doact_title(ofl->ofl_lml));

768         /*
769          * Process active relocations.
770          */
771         REL_CACHE_TRAVERSE(&ofl->ofl_actrels, idx, rcbp, arsp) {
772                 uchar_t         *addr;
773                 Xword           value;
774                 Sym_desc        *sdp;
775                 const char      *ifl_name;
776                 Xword           refaddr;
777                 int             moved = 0;
778                 Gotref          gref;
779                 Os_desc         *osp;

781                 /*
782                  * If the section this relocation is against has been discarded
783                  * (-zignore), then discard (skip) the relocation itself.
784                  */
785                 if ((arsp->rel_isdesc->is_flags & FLG_IS_DISCARD) &&
786                     ((arsp->rel_flags & (FLG_REL_GOT | FLG_REL_BSS |
787                     FLG_REL_PLT | FLG_REL_NOINFO)) == 0)) {
788                         DBG_CALL(Dbg_reloc_discard(ofl->ofl_lml, M_MACH, arsp));
789                         continue;
790                 }

792                 /*
793                  * We determine what the 'got reference' model (if required)
794                  * is at this point.  This needs to be done before tls_fixup()
795                  * since it may 'transition' our instructions.
796                  *
797                  * The got table entries have already been assigned,
798                  * and we bind to those initial entries.
799                  */
800                 if (arsp->rel_flags & FLG_REL_DTLS)
801                         gref = GOT_REF_TLSGD;
802                 else if (arsp->rel_flags & FLG_REL_MTLS)
803                         gref = GOT_REF_TLSLD;
804                 else if (arsp->rel_flags & FLG_REL_STLS)
805                         gref = GOT_REF_TLSIE;
806                 else
807                         gref = GOT_REF_GENERIC;

809                 /*
810                  * Perform any required TLS fixups.
811                  */
812                 if (arsp->rel_flags & FLG_REL_TLSFIX) {
813                         Fixupret        ret;

815                         if ((ret = tls_fixups(ofl, arsp)) == FIX_ERROR)
816                                 return (S_ERROR);
817                         if (ret == FIX_DONE)
818                                 continue;
819                 }

821                 /*
822                  * If this is a relocation against a move table, or
823                  * expanded move table, adjust the relocation entries.
824                  */
825                 if (RELAUX_GET_MOVE(arsp))
826                         ld_adj_movereloc(ofl, arsp);
```



```
828                 sdp = arsp->rel_sym;
829                 refaddr = arsp->rel_roffset +
830                     (Off)_elf_getxoff(arsp->rel_isdesc->is_indata);

832                 if ((arsp->rel_flags & FLG_REL_CLVAL) ||
833                     (arsp->rel_flags & FLG_REL_GOTCL))
834                         value = 0;
835                 else if (ELF_ST_TYPE(sdp->sd_sym->st_info) == STT_SECTION) {
836                         Sym_desc        *sym;

838                         /*
839                          * The value for a symbol pointing to a SECTION
840                          * is based off of that sections position.
841                          */
842                         if ((sdp->sd_isc->is_flags & FLG_IS_RELUPD) &&
843                             /* LINTED */
844                             (sym = ld_am_I_partial(arsp, arsp->rel_raddend))) {
845                                 /*
846                                  * The symbol was moved, so adjust the value
847                                  * relative to the new section.
848                                  */
849                                 value = sym->sd_sym->st_value;
850                                 moved = 1;

852                                 /*
853                                  * The original raddend covers the displacement
854                                  * from the section start to the desired
855                                  * address. The value computed above gets us
856                                  * from the section start to the start of the
857                                  * symbol range. Adjust the old raddend to
858                                  * remove the offset from section start to
859                                  * symbol start, leaving the displacement
860                                  * within the range of the symbol.
861                                  */
862                                 arsp->rel_raddend -= sym->sd_osym->st_value;
863                         } else {
864                                 value = _elf_getxoff(sdp->sd_isc->is_indata);
865                                 if (sdp->sd_isc->is_shdr->sh_flags & SHF_ALLOC)
866                                         value += sdp->sd_isc->is_osdesc->
867                                             os_shdr->sh_addr;
868                         }
869                         if (sdp->sd_isc->is_shdr->sh_flags & SHF_TLS)
870                                 value -= ofl->ofl_tlsphdr->p_vaddr;

872                 } else if (IS_SIZE(arsp->rel_rtype)) {
873                         /*
874                          * Size relocations require the symbols size.
875                          */
876                         value = sdp->sd_sym->st_size;

878                 } else if ((sdp->sd_flags & FLG_SY_CAP) &&
879                     sdp->sd_aux && sdp->sd_aux->sa_PLTndx) {
880                         /*
881                          * If relocation is against a capabilities symbol, we
882                          * need to jump to an associated PLT, so that at runtime
883                          * ld.so.1 is involved to determine the best binding
884                          * choice. Otherwise, the value is the symbols value.
885                          */
886                         value = ld_calc_plt_addr(sdp, ofl);
887                 } else
888                         value = sdp->sd_sym->st_value;

890                 /*
891                  * Relocation against the GLOBAL_OFFSET_TABLE.
892                  */
```

```
893                 if ((arsp->rel_flags & FLG_REL_GOT) &&
894                     !ld_reloc_set_aux_osdesc(ofl, arsp, ofl->ofl_osgot))
895                         return (S_ERROR);
896                 osp = RELAUX_GET_OSDESC(arsp);
897
898                 /*
899                  * If loadable and not producing a relocatable object add the
900                  * sections virtual address to the reference address.
901                  */
902                 if ((arsp->rel_flags & FLG_REL_LOAD) &&
903                     ((flags & FLG_OF_RELOBJ) == 0))
904                         refaddr += arsp->rel_isdesc->is_osdesc->
905                             os_shdr->sh_addr;
906
907                 /*
908                  * If this entry has a PLT assigned to it, its value is actually
909                  * the address of the PLT (and not the address of the function).
910                  */
911                 if (IS_PLT(arsp->rel_rtype)) {
912                         if (sdp->sd_aux && sdp->sd_aux->sa_PLTndx)
913                                 value = ld_calc_plt_addr(sdp, ofl);
914                 }
915
916                 /*
917                  * Add relocations addend to value.  Add extra
918                  * relocation addend if needed.
919                  *
920                  * Note: For GOT relative relocations on amd64 we discard the
921                  * addend.  It was relevant to the reference - not to the
922                  * data item being referenced (ie: that -4 thing).
923                  */
924                 if ((arsp->rel_flags & FLG_REL_GOT) == 0)
925                         value += arsp->rel_raddend;
926
927                 /*
928                  * Determine whether the value needs further adjustment. Filter
929                  * through the attributes of the relocation to determine what
930                  * adjustment is required.  Note, many of the following cases
931                  * are only applicable when a .got is present.  As a .got is
932                  * not generated when a relocatable object is being built,
933                  * any adjustments that require a .got need to be skipped.
934                  */
935                 if ((arsp->rel_flags & FLG_REL_GOT) &&
936                     ((flags & FLG_OF_RELOBJ) == 0)) {
937                         Xword           R1addr;
938                         uintptr_t       R2addr;
939                         Word            gotndx;
940                         Gotndx          *gnp;
941
942                         /*
943                          * Perform relocation against GOT table. Since this
944                          * doesn't fit exactly into a relocation we place the
945                          * appropriate byte in the GOT directly
946                          *
947                          * Calculate offset into GOT at which to apply
948                          * the relocation.
949                          */
950                         gnp = ld_find_got_ndx(sdp->sd_GOTndxs, gref, ofl, arsp);
951                         assert(gnp);
952
953                         if (arsp->rel_rtype == R_AMD64_DTPOFF64)
954                                 gotndx = gnp->gn_gotndx + 1;
955                         else
956                                 gotndx = gnp->gn_gotndx;
957
958                         R1addr = (Xword)(gotndx * M_GOT_ENTSIZE);
```



```
960                         /*
961                          * Add the GOTs data's offset.
962                          */
963                         R2addr = R1addr + (uintptr_t)osp->os_outdata->d_buf;
964
965                         DBG_CALL(Dbg_reloc_doact(ofl->ofl_lml, ELF_DBG_LD_ACT,
966                             M_MACH, SHT_RELA, arsp, R1addr, value,
967                             ld_reloc_sym_name));
968
969                         /*
970                          * And do it.
971                          */
972                         if (ofl->ofl_flags1 & FLG_OF1_ENCDIFF)
973                                 *(Xword *)R2addr = ld_bswap_Xword(value);
974                         else
975                                 *(Xword *)R2addr = value;
976                         continue;
977
978                 } else if (IS_GOT_BASED(arsp->rel_rtype) &&
979                     ((flags & FLG_OF_RELOBJ) == 0)) {
980                         value -= ofl->ofl_osgot->os_shdr->sh_addr;
981
982                 } else if (IS_GOTPCREL(arsp->rel_rtype) &&
983                     ((flags & FLG_OF_RELOBJ) == 0)) {
984                         Gotndx *gnp;
985
986                         /*
987                          * Calculation:
988                          *      G + GOT + A - P
989                          */
990                         gnp = ld_find_got_ndx(sdp->sd_GOTndxs, gref, ofl, arsp);
991                         assert(gnp);
992                         value = (Xword)(ofl->ofl_osgot->os_shdr-> sh_addr) +
993                             ((Xword)gnp->gn_gotndx * M_GOT_ENTSIZE) +
994                             arsp->rel_raddend - refaddr;
995
996                 } else if (IS_GOT_PC(arsp->rel_rtype) &&
997                     ((flags & FLG_OF_RELOBJ) == 0)) {
998                         value = (Xword)(ofl->ofl_osgot->os_shdr->
999                             sh_addr) - refaddr + arsp->rel_raddend;
1000
1001                } else if ((IS_PC_RELATIVE(arsp->rel_rtype)) &&
1002                    (((flags & FLG_OF_RELOBJ) == 0) ||
1003                    (osp == sdp->sd_isc->is_osdesc))) {
1004                        value -= refaddr;
1005
1006                } else if (IS_TLS_INS(arsp->rel_rtype) &&
1007                    IS_GOT_RELATIVE(arsp->rel_rtype) &&
1008                    ((flags & FLG_OF_RELOBJ) == 0)) {
1009                        Gotndx  *gnp;
1010
1011                        gnp = ld_find_got_ndx(sdp->sd_GOTndxs, gref, ofl, arsp);
1012                        assert(gnp);
1013                        value = (Xword)gnp->gn_gotndx * M_GOT_ENTSIZE;
1014
1015                } else if (IS_GOT_RELATIVE(arsp->rel_rtype) &&
1016                    ((flags & FLG_OF_RELOBJ) == 0)) {
1017                        Gotndx *gnp;
1018
1019                        gnp = ld_find_got_ndx(sdp->sd_GOTndxs, gref, ofl, arsp);
1020                        assert(gnp);
1021                        value = (Xword)gnp->gn_gotndx * M_GOT_ENTSIZE;
1022
1023                } else if ((arsp->rel_flags & FLG_REL_STLS) &&
1024                    ((flags & FLG_OF_RELOBJ) == 0)) {
```

```
1025                    Xword   tlsstatsize;
1027                    /*
1028                     * This is the LE TLS reference model.  Static
1029                     * offset is hard-coded.
1030                     */
1031                    tlsstatsize = S_ROUND(ofl->ofl_tlsphdr->p_memsz,
1032                        M_TLSSTATALIGN);
1033                    value = tlsstatsize - value;
1035                    /*
1036                     * Since this code is fixed up, it assumes a negative
1037                     * offset that can be added to the thread pointer.
1038                     */
1039                    if (arsp->rel_rtype == R_AMD64_TPOFF32)
1040                            value = -value;
1041            }
1043            if (arsp->rel_isdesc->is_file)
1044                    ifl_name = arsp->rel_isdesc->is_file->ifl_name;
1045            else
1046                    ifl_name = MSG_INTL(MSG_STR_NULL);
1048            /*
1049             * Make sure we have data to relocate.  Compiler and assembler
1050             * developers have been known to generate relocations against
1051             * invalid sections (normally .bss), so for their benefit give
1052             * them sufficient information to help analyze the problem.
1053             * End users should never see this.
1054             */
1055            if (arsp->rel_isdesc->is_indata->d_buf == 0) {
1056                    Conv_inv_buf_t inv_buf;
1058                    ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_REL_EMPTYSEC),
1059                        conv_reloc_amd64_type(arsp->rel_rtype, 0, &inv_buf),
1060                        ifl_name, ld_reloc_sym_name(arsp),
1061                        EC_WORD(arsp->rel_isdesc->is_scnndx),
1062                        arsp->rel_isdesc->is_name);
1063                    return (S_ERROR);
1064            }
1066            /*
1067             * Get the address of the data item we need to modify.
1068             */
1069            addr = (uchar_t *)((uintptr_t)arsp->rel_roffset +
1070                (uintptr_t)_elf_getxoff(arsp->rel_isdesc->is_indata));
1072            DBG_CALL(Dbg_reloc_doact(ofl->ofl_lml, ELF_DBG_LD_ACT,
1073                M_MACH, SHT_RELA, arsp, EC_NATPTR(addr), value,
1074                ld_reloc_sym_name));
1075            addr += (uintptr_t)osp->os_outdata->d_buf;
1077            if ((((uintptr_t)addr - (uintptr_t)ofl->ofl_nehdr) >
1078                ofl->ofl_size) || (arsp->rel_roffset >
1079                osp->os_shdr->sh_size)) {
1080                    int             class;
1081                    Conv_inv_buf_t inv_buf;
1083                    if (((uintptr_t)addr - (uintptr_t)ofl->ofl_nehdr) >
1084                        ofl->ofl_size)
1085                            class = ERR_FATAL;
1086                    else
1087                            class = ERR_WARNING;
1089                    ld_eprintf(ofl, class, MSG_INTL(MSG_REL_INVALOFFSET),
1090                        conv_reloc_amd64_type(arsp->rel_rtype, 0, &inv_buf),
```



```
1091                            ifl_name, EC_WORD(arsp->rel_isdesc->is_scnndx),
1092                            arsp->rel_isdesc->is_name, ld_reloc_sym_name(arsp),
1093                            EC_ADDR((uintptr_t)addr -
1094                            (uintptr_t)ofl->ofl_nehdr));
1096                    if (class == ERR_FATAL) {
1097                            return_code = S_ERROR;
1098                            continue;
1099                    }
1100            }
1102            /*
1103             * The relocation is additive.  Ignore the previous symbol
1104             * value if this local partial symbol is expanded.
1105             */
1106            if (moved)
1107                    value -= *addr;
1109            /*
1110             * If '-z noreloc' is specified - skip the do_reloc_ld stage.
1111             */
1112            if (OFL_DO_RELOC(ofl)) {
1113                    /*
1114                     * If this is a PROGBITS section and the running linker
1115                     * has a different byte order than the target host,
1116                     * tell do_reloc_ld() to swap bytes.
1117                     */
1118                    if (do_reloc_ld(arsp, addr, &value, ld_reloc_sym_name,
1119                        ifl_name, OFL_SWAP_RELOC_DATA(ofl, arsp),
1120                        ofl->ofl_lml) == 0) {
1121                            ofl->ofl_flags |= FLG_OF_FATAL;
1122                            return_code = S_ERROR;
1123                    }
1124            }
1125    }
1126    return (return_code);
1127 }
1129 static uintptr_t
1130 ld_add_outrel(Word flags, Rel_desc *rsp, Ofl_desc *ofl)
1131 {
1132    Rel_desc        *orsp;
1133    Sym_desc        *sdp = rsp->rel_sym;
1135    /*
1136     * Static executables *do not* want any relocations against them.
1137     * Since our engine still creates relocations against a WEAK UNDEFINED
1138     * symbol in a static executable, it's best to disable them here
1139     * instead of through out the relocation code.
1140     */
1141    if (OFL_IS_STATIC_EXEC(ofl))
1142            return (1);
1144    /*
1145     * If we are adding a output relocation against a section
1146     * symbol (non-RELATIVE) then mark that section.  These sections
1147     * will be added to the .dynsym symbol table.
1148     */
1149    if (sdp && (rsp->rel_rtype != M_R_RELATIVE) &&
1150        ((flags & FLG_REL_SCNNDX) ||
1151        (ELF_ST_TYPE(sdp->sd_sym->st_info) == STT_SECTION))) {
1153            /*
1154             * If this is a COMMON symbol - no output section
1155             * exists yet - (it's created as part of sym_validate()).
1156             * So - we mark here that when it's created it should
```

```
1157                 * be tagged with the FLG_OS_OUTREL flag.
1158                 */
1159                if ((sdp->sd_flags & FLG_SY_SPECSEC) &&
1160                    (sdp->sd_sym->st_shndx == SHN_COMMON)) {
1161                        if (ELF_ST_TYPE(sdp->sd_sym->st_info) != STT_TLS)
1162                                ofl->ofl_flags1 |= FLG_OF1_BSSOREL;
1163                        else
1164                                ofl->ofl_flags1 |= FLG_OF1_TLSOREL;
1165                } else {
1166                        Os_desc *osp;
1167                        Is_desc *isp = sdp->sd_isc;

1169                        if (isp && ((osp = isp->is_osdesc) != NULL) &&
1170                            ((osp->os_flags & FLG_OS_OUTREL) == 0)) {
1171                                ofl->ofl_dynshdrcnt++;
1172                                osp->os_flags |= FLG_OS_OUTREL;
1173                        }
1174                }
1175        }

1177        /* Enter it into the output relocation cache */
1178        if ((orsp = ld_reloc_enter(ofl, &ofl->ofl_outrels, rsp, flags)) == NULL)
1179                return (S_ERROR);

1181        if (flags & FLG_REL_GOT)
1182                ofl->ofl_relocgotsz += (Xword)sizeof (Rela);
1183        else if (flags & FLG_REL_PLT)
1184                ofl->ofl_relocpltsz += (Xword)sizeof (Rela);
1185        else if (flags & FLG_REL_BSS)
1186                ofl->ofl_relocbsssz += (Xword)sizeof (Rela);
1187        else if (flags & FLG_REL_NOINFO)
1188                ofl->ofl_relocrelsz += (Xword)sizeof (Rela);
1189        else
1190                RELAUX_GET_OSDESC(orsp)->os_szoutrels += (Xword)sizeof (Rela);

1192        if (orsp->rel_rtype == M_R_RELATIVE)
1193                ofl->ofl_relocrelcnt++;

1195        /*
1196         * We don't perform sorting on PLT relocations because
1197         * they have already been assigned a PLT index and if we
1198         * were to sort them we would have to re-assign the plt indexes.
1199         */
1200        if (!(flags & FLG_REL_PLT))
1201                ofl->ofl_reloccnt++;

1203        /*
1204         * Insure a GLOBAL_OFFSET_TABLE is generated if required.
1205         */
1206        if (IS_GOT_REQUIRED(orsp->rel_rtype))
1207                ofl->ofl_flags |= FLG_OF_BLDGOT;

1209        /*
1210         * Identify and possibly warn of a displacement relocation.
1211         */
1212        if (orsp->rel_flags & FLG_REL_DISP) {
1213                ofl->ofl_dtflags_1 |= DF_1_DISPRELPND;

1215                if (ofl->ofl_flags & FLG_OF_VERBOSE)
1216                        ld_disp_errmsg(MSG_INTL(MSG_REL_DISPREL4), orsp, ofl);
1217        }
1218        DBG_CALL(Dbg_reloc_ors_entry(ofl->ofl_lml, ELF_DBG_LD, SHT_RELA,
1219            M_MACH, orsp));
1220        return (1);
1221 }
```



```
1223 /*
1224  * process relocation for a LOCAL symbol
1225  */
1226 static uintptr_t
1227 ld_reloc_local(Rel_desc * rsp, Ofl_desc * ofl)
1228 {
1229        ofl_flag_t      flags = ofl->ofl_flags;
1230        Sym_desc        *sdp = rsp->rel_sym;
1231        Word            shndx = sdp->sd_sym->st_shndx;
1232        Word            ortype = rsp->rel_rtype;

1234        /*
1235         * if ((shared object) and (not pc relative relocation) and
1236         *    (not against ABS symbol))
1237         * then
1238         *      build R_AMD64_RELATIVE
1239         * fi
1240         */
1241        if ((flags & FLG_OF_SHAROBJ) && (rsp->rel_flags & FLG_REL_LOAD) &&
1242            !(IS_PC_RELATIVE(rsp->rel_rtype)) && !(IS_SIZE(rsp->rel_rtype)) &&
1243            !(IS_GOT_BASED(rsp->rel_rtype)) &&
1244            !(rsp->rel_isdesc != NULL &&
1245            (rsp->rel_isdesc->is_shdr->sh_type == SHT_SUNW_dof)) &&
1246            (((sdp->sd_flags & FLG_SY_SPECSEC) == 0) ||
1247            (shndx != SHN_ABS) || (sdp->sd_aux && sdp->sd_aux->sa_symspec))) {

1249                /*
1250                 * R_AMD64_RELATIVE updates a 64bit address, if this
1251                 * relocation isn't a 64bit binding then we can not
1252                 * simplify it to a RELATIVE relocation.
1253                 */
1254                if (reloc_table[ortype].re_fsize != sizeof (Addr)) {
1255                        return (ld_add_outrel(0, rsp, ofl));
1256                }

1258                rsp->rel_rtype = R_AMD64_RELATIVE;
1259                if (ld_add_outrel(FLG_REL_ADVAL, rsp, ofl) == S_ERROR)
1260                        return (S_ERROR);
1261                rsp->rel_rtype = ortype;
1262                return (1);
1263        }

1265        /*
1266         * If the relocation is against a 'non-allocatable' section
1267         * and we can not resolve it now - then give a warning
1268         * message.
1269         *
1270         * We can not resolve the symbol if either:
1271         *      a) it's undefined
1272         *      b) it's defined in a shared library and a
1273         *         COPY relocation hasn't moved it to the executable
1274         *
1275         * Note: because we process all of the relocations against the
1276         *      text segment before any others - we know whether
1277         *      or not a copy relocation will be generated before
1278         *      we get here (see reloc_init()->reloc_segments()).
1279         */
1280        if (!(rsp->rel_flags & FLG_REL_LOAD) &&
1281            ((shndx == SHN_UNDEF) ||
1282            ((sdp->sd_ref == REF_DYN_NEED) &&
1283            ((sdp->sd_flags & FLG_SY_MVTOCOMM) == 0)))) {
1284                Conv_inv_buf_t  inv_buf;
1285                Os_desc         *osp = RELAUX_GET_OSDESC(rsp);

1287                /*
1288                 * If the relocation is against a SHT_SUNW_ANNOTATE
```

```
1289                  * section - then silently ignore that the relocation
1290                  * can not be resolved.
1291                  */
1292                 if (osp && (osp->os_shdr->sh_type == SHT_SUNW_ANNOTATE))
1293                         return (0);
1294                 ld_eprintf(ofl, ERR_WARNING, MSG_INTL(MSG_REL_EXTERNSYM),
1295                     conv_reloc_amd64_type(rsp->rel_rtype, 0, &inv_buf),
1296                     rsp->rel_isdesc->is_file->ifl_name,
1297                     ld_reloc_sym_name(rsp), osp->os_name);
1298                 return (1);
1299         }

1301         /*
1302          * Perform relocation.
1303          */
1304         return (ld_add_actrel(NULL, rsp, ofl));
1305 }


1308 static uintptr_t
1309 ld_reloc_TLS(Boolean local, Rel_desc * rsp, Ofl_desc * ofl)
1310 {
1311         Word            rtype = rsp->rel_rtype;
1312         Sym_desc        *sdp = rsp->rel_sym;
1313         ofl_flag_t      flags = ofl->ofl_flags;
1314         Gotndx          *gnp;

1316         /*
1317          * If we're building an executable - use either the IE or LE access
1318          * model.  If we're building a shared object process any IE model.
1319          */
1320         if ((flags & FLG_OF_EXEC) || (IS_TLS_IE(rtype))) {
1321                 /*
1322                  * Set the DF_STATIC_TLS flag.
1323                  */
1324                 ofl->ofl_dtflags |= DF_STATIC_TLS;

1326                 if (!local || ((flags & FLG_OF_EXEC) == 0)) {
1327                         /*
1328                          * Assign a GOT entry for static TLS references.
1329                          */
1330                         if ((gnp = ld_find_got_ndx(sdp->sd_GOTndxs,
1331                             GOT_REF_TLSIE, ofl, rsp)) == NULL) {

1333                                 if (ld_assign_got_TLS(local, rsp, ofl, sdp,
1334                                     gnp, GOT_REF_TLSIE, FLG_REL_STLS,
1335                                     rtype, R_AMD64_TPOFF64, 0) == S_ERROR)
1336                                         return (S_ERROR);
1337                         }

1339                         /*
1340                          * IE access model.
1341                          */
1342                         if (IS_TLS_IE(rtype))
1343                                 return (ld_add_actrel(FLG_REL_STLS, rsp, ofl));

1345                         /*
1346                          * Fixups are required for other executable models.
1347                          */
1348                         return (ld_add_actrel((FLG_REL_TLSFIX | FLG_REL_STLS),
1349                             rsp, ofl));
1350                 }

1352                 /*
1353                  * LE access model.
1354                  */
```



```
1355                 if (IS_TLS_LE(rtype))
1356                         return (ld_add_actrel(FLG_REL_STLS, rsp, ofl));

1358                 return (ld_add_actrel((FLG_REL_TLSFIX | FLG_REL_STLS),
1359                     rsp, ofl));
1360         }

1362         /*
1363          * Building a shared object.
1364          *
1365          * Assign a GOT entry for a dynamic TLS reference.
1366          */
1367         if (IS_TLS_LD(rtype) && ((gnp = ld_find_got_ndx(sdp->sd_GOTndxs,
1368             GOT_REF_TLSLD, ofl, rsp)) == NULL)) {

1370                 if (ld_assign_got_TLS(local, rsp, ofl, sdp, gnp, GOT_REF_TLSLD,
1371                     FLG_REL_MTLS, rtype, R_AMD64_DTPMOD64, NULL) == S_ERROR)
1372                         return (S_ERROR);

1374         } else if (IS_TLS_GD(rtype) &&
1375             ((gnp = ld_find_got_ndx(sdp->sd_GOTndxs, GOT_REF_TLSGD,
1376             ofl, rsp)) == NULL)) {

1378                 if (ld_assign_got_TLS(local, rsp, ofl, sdp, gnp, GOT_REF_TLSGD,
1379                     FLG_REL_DTLS, rtype, R_AMD64_DTPMOD64,
1380                     R_AMD64_DTPOFF64) == S_ERROR)
1381                         return (S_ERROR);
1382         }

1384         if (IS_TLS_LD(rtype))
1385                 return (ld_add_actrel(FLG_REL_MTLS, rsp, ofl));

1387         return (ld_add_actrel(FLG_REL_DTLS, rsp, ofl));
1388 }

1390 /* ARGSUSED5 */
1391 static uintptr_t
1392 ld_assign_got_ndx(Alist **alpp, Gotndx *pgnp, Gotref gref, Ofl_desc *ofl,
1393     Rel_desc *rsp, Sym_desc *sdp)
1394 {
1395         Xword           raddend;
1396         Gotndx          gn, *gnp;
1397         Aliste          idx;
1398         uint_t          gotents;

1400         raddend = rsp->rel_raddend;
1401         if (pgnp && (pgnp->gn_addend == raddend) && (pgnp->gn_gotref == gref))
1402                 return (1);

1404         if ((gref == GOT_REF_TLSGD) || (gref == GOT_REF_TLSLD))
1405                 gotents = 2;
1406         else
1407                 gotents = 1;

1409         gn.gn_addend = raddend;
1410         gn.gn_gotndx = ofl->ofl_gotcnt;
1411         gn.gn_gotref = gref;

1413         ofl->ofl_gotcnt += gotents;

1415         if (gref == GOT_REF_TLSLD) {
1416                 if (ofl->ofl_tlsldgotndx == NULL) {
1417                         if ((gnp = libld_malloc(sizeof (Gotndx))) == NULL)
1418                                 return (S_ERROR);
1419                         (void) memcpy(gnp, &gn, sizeof (Gotndx));
1420                         ofl->ofl_tlsldgotndx = gnp;
```

```
1421                }
1422                return (1);
1423        }

1425        idx = 0;
1426        for (ALIST_TRAVERSE(*alpp, idx, gnp)) {
1427                if (gnp->gn_addend > raddend)
1428                        break;
1429        }

1431        /*
1432         * GOT indexes are maintained on an Alist, where there is typically
1433         * only one index.  The usage of this list is to scan the list to find
1434         * an index, and then apply that index immediately to a relocation.
1435         * Thus there are no external references to these GOT index structures
1436         * that can be compromised by the Alist being reallocated.
1437         */
1438        if (alist_insert(alpp, &gn, sizeof (Gotndx),
1439            AL_CNT_SDP_GOT, idx) == NULL)
1440                return (S_ERROR);

1442        return (1);
1443 }

1445 static void
1446 ld_assign_plt_ndx(Sym_desc * sdp, Ofl_desc *ofl)
1447 {
1448        sdp->sd_aux->sa_PLTndx = 1 + ofl->ofl_pltcnt++;
1449        sdp->sd_aux->sa_PLTGOTndx = ofl->ofl_gotcnt++;
1450        ofl->ofl_flags |= FLG_OF_BLDGOT;
1451 }

1453 static uchar_t plt0_template[M_PLT_ENTSIZE] = {
1454 /* 0x00 PUSHQ GOT+8(%rip) */    0xff, 0x35, 0x00, 0x00, 0x00, 0x00,
1455 /* 0x06 JMP   *GOT+16(%rip) */  0xff, 0x25, 0x00, 0x00, 0x00, 0x00,
1456 /* 0x0c NOP */                  0x90,
1457 /* 0x0d NOP */                  0x90,
1458 /* 0x0e NOP */                  0x90,
1459 /* 0x0f NOP */                  0x90
1460 };

1462 /*
1463  * Initializes .got[0] with the _DYNAMIC symbol value.
1464  */
1465 static uintptr_t
1466 ld_fillin_gotplt(Ofl_desc *ofl)
1467 {
1468        int     bswap = (ofl->ofl_flags1 & FLG_OF1_ENCDIFF) != 0;

1470        if (ofl->ofl_osgot) {
1471                Sym_desc        *sdp;

1473                if ((sdp = ld_sym_find(MSG_ORIG(MSG_SYM_DYNAMIC_U),
1474                    SYM_NOHASH, NULL, ofl)) != NULL) {
1475                        uchar_t *genptr;

1477                        genptr = ((uchar_t *)ofl->ofl_osgot->os_outdata->d_buf +
1478                            (M_GOT_XDYNAMIC * M_GOT_ENTSIZE));
1479                        /* LINTED */
1480                        *(Xword *)genptr = sdp->sd_sym->st_value;
1481                        if (bswap)
1482                                /* LINTED */
1483                                *(Xword *)genptr =
1484                                    /* LINTED */
1485                                    ld_bswap_Xword(*(Xword *)genptr);
1486                }
```



```
1487        }

1489        /*
1490         * Fill in the reserved slot in the procedure linkage table the first
1491         * entry is:
1492         *      0x00 PUSHQ      GOT+8(%rip)         # GOT[1]
1493         *      0x06 JMP        *GOT+16(%rip)       # GOT[2]
1494         *      0x0c NOP
1495         *      0x0d NOP
1496         *      0x0e NOP
1497         *      0x0f NOP
1498         */
1499        if ((ofl->ofl_flags & FLG_OF_DYNAMIC) && ofl->ofl_osplt) {
1500                uchar_t *pltent;
1501                Xword   val1;

1503                pltent = (uchar_t *)ofl->ofl_osplt->os_outdata->d_buf;
1504                bcopy(plt0_template, pltent, sizeof (plt0_template));

1506                /*
1507                 * If '-z noreloc' is specified - skip the do_reloc_ld
1508                 * stage.
1509                 */
1510                if (!OFL_DO_RELOC(ofl))
1511                        return (1);

1513                /*
1514                 * filin:
1515                 *      PUSHQ GOT + 8(%rip)
1516                 *
1517                 * Note: 0x06 below represents the offset to the
1518                 *       next instruction - which is what %rip will
1519                 *       be pointing at.
1520                 */
1521                val1 = (ofl->ofl_osgot->os_shdr->sh_addr) +
1522                    (M_GOT_XLINKMAP * M_GOT_ENTSIZE) -
1523                    ofl->ofl_osplt->os_shdr->sh_addr - 0x06;

1525                if (do_reloc_ld(&rdesc_r_amd64_gotpcrel, &pltent[0x02],
1526                    &val1, syn_rdesc_sym_name, MSG_ORIG(MSG_SPECFIL_PLTENT),
1527                    bswap, ofl->ofl_lml) == 0) {
1528                        ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_PLT_PLT0FAIL));
1529                        return (S_ERROR);
1530                }

1532                /*
1533                 * filin:
1534                 *  JMP *GOT+16(%rip)
1535                 */
1536                val1 = (ofl->ofl_osgot->os_shdr->sh_addr) +
1537                    (M_GOT_XRTLD * M_GOT_ENTSIZE) -
1538                    ofl->ofl_osplt->os_shdr->sh_addr - 0x0c;

1540                if (do_reloc_ld(&rdesc_r_amd64_gotpcrel, &pltent[0x08],
1541                    &val1, syn_rdesc_sym_name, MSG_ORIG(MSG_SPECFIL_PLTENT),
1542                    bswap, ofl->ofl_lml) == 0) {
1543                        ld_eprintf(ofl, ERR_FATAL, MSG_INTL(MSG_PLT_PLT0FAIL));
1544                        return (S_ERROR);
1545                }
1546        }

1548        return (1);
1549 }
```

```
1553 /*
1554  * Template for generating "void (*)(void)" function
1555  */
1556 static const uchar_t nullfunc_tmpl[] = {        /* amd64 */
1557 /* 0x00 */      0x55,                           /* pushq  %rbp */
1558 /* 0x01 */      0x48, 0x8b, 0xec,               /* movq   %rsp,%rbp */
1559 /* 0x04 */      0x48, 0x8b, 0xe5,               /* movq   %rbp,%rsp */
1560 /* 0x07 */      0x5d,                           /* popq   %rbp */
1561 /* 0x08 */      0xc3                            /* ret */
1562 };


1565 /*
1566  * Function used to provide fill padding in SHF_EXECINSTR sections
1567  *
1568  * entry:
1569  *
1570  *      base - base address of section being filled
1571  *      offset - starting offset for fill within memory referenced by base
1572  *      cnt - # bytes to be filled
1573  *
1574  * exit:
1575  *      The fill has been completed.
1576  */
1577 static void
1578 execfill(void *base, off_t off, size_t cnt)
1579 {
1580         /*
1581          * 0x90 is an X86 NOP instruction in both 32 and 64-bit worlds.
1582          * There are no alignment constraints.
1583          */
1584         (void) memset(off + (char *)base, 0x90, cnt);
1585 }


1588 /*
1589  * Return the ld_targ definition for this target.
1590  */
1591 const Target *
1592 ld_targ_init_x86(void)
1593 {
1594         static const Target _ld_targ = {
1595                 {                       /* Target_mach */
1596                         M_MACH,                 /* m_mach */
1597                         M_MACHPLUS,             /* m_machplus */
1598                         M_FLAGSPLUS,            /* m_flagsplus */
1599                         M_CLASS,                /* m_class */
1600                         M_DATA,                 /* m_data */

1602                         M_SEGM_ALIGN,           /* m_segm_align */
1603                         M_SEGM_ORIGIN,          /* m_segm_origin */
1604                         M_SEGM_AORIGIN,         /* m_segm_aorigin */
1605                         M_DATASEG_PERM,         /* m_dataseg_perm */
1606                         M_STACK_PERM,           /* m_stack_perm */
1607                         M_WORD_ALIGN,           /* m_word_align */
1608                         MSG_ORIG(MSG_PTH_RTLD_AMD64), /* m_def_interp */

1610                         /* Relocation type codes */
1611                         M_R_ARRAYADDR,          /* m_r_arrayaddr */
1612                         M_R_COPY,               /* m_r_copy */
1613                         M_R_GLOB_DAT,           /* m_r_glob_dat */
1614                         M_R_JMP_SLOT,           /* m_r_jmp_slot */
1615                         M_R_NUM,                /* m_r_num */
1616                         M_R_NONE,               /* m_r_none */
1617                         M_R_RELATIVE,           /* m_r_relative */
1618                         M_R_REGISTER,           /* m_r_register */
```



```
1620                         /* Relocation related constants */
1621                         M_REL_DT_COUNT,         /* m_rel_dt_count */
1622                         M_REL_DT_ENT,           /* m_rel_dt_ent */
1623                         M_REL_DT_SIZE,          /* m_rel_dt_size */
1624                         M_REL_DT_TYPE,          /* m_rel_dt_type */
1625                         M_REL_SHT_TYPE,         /* m_rel_sht_type */

1627                         /* GOT related constants */
1628                         M_GOT_ENTSIZE,          /* m_got_entsize */
1629                         M_GOT_XNumber,          /* m_got_xnumber */

1631                         /* PLT related constants */
1632                         M_PLT_ALIGN,            /* m_plt_align */
1633                         M_PLT_ENTSIZE,          /* m_plt_entsize */
1634                         M_PLT_RESERVSZ,         /* m_plt_reservsz */
1635                         M_PLT_SHF_FLAGS,        /* m_plt_shf_flags */

1637                         /* Section type of .eh_frame/.eh_frame_hdr sections */
1638                         SHT_AMD64_UNWIND,       /* m_sht_unwind */

1640                         M_DT_REGISTER,          /* m_dt_register */
1641                 },
1642                 {                               /* Target_machid */
1643                         M_ID_ARRAY,             /* id_array */
1644                         M_ID_BSS,               /* id_bss */
1645                         M_ID_CAP,               /* id_cap */
1646                         M_ID_CAPINFO,           /* id_capinfo */
1647                         M_ID_CAPCHAIN,          /* id_capchain */
1648                         M_ID_DATA,              /* id_data */
1649                         M_ID_DYNAMIC,           /* id_dynamic */
1650                         M_ID_DYNSORT,           /* id_dynsort */
1651                         M_ID_DYNSTR,            /* id_dynstr */
1652                         M_ID_DYNSYM,            /* id_dynsym */
1653                         M_ID_DYNSYM_NDX,        /* id_dynsym_ndx */
1654                         M_ID_GOT,               /* id_got */
1655                         M_ID_UNKNOWN,           /* id_gotdata (unused) */
1656                         M_ID_HASH,              /* id_hash */
1657                         M_ID_INTERP,            /* id_interp */
1658                         M_ID_LBSS,              /* id_lbss */
1659                         M_ID_LDYNSYM,           /* id_ldynsym */
1660                         M_ID_NOTE,              /* id_note */
1661                         M_ID_NULL,              /* id_null */
1662                         M_ID_PLT,               /* id_plt */
1663                         M_ID_REL,               /* id_rel */
1664                         M_ID_STRTAB,            /* id_strtab */
1665                         M_ID_SYMINFO,           /* id_syminfo */
1666                         M_ID_SYMTAB,            /* id_symtab */
1667                         M_ID_SYMTAB_NDX,        /* id_symtab_ndx */
1668                         M_ID_TEXT,              /* id_text */
1669                         M_ID_TLS,               /* id_tls */
1670                         M_ID_TLSBSS,            /* id_tlsbss */
1671                         M_ID_UNKNOWN,           /* id_unknown */
1672                         M_ID_UNWIND,            /* id_unwind */
1673                         M_ID_UNWINDHDR,         /* id_unwindhdr */
1674                         M_ID_USER,              /* id_user */
1675                         M_ID_VERSION,           /* id_version */
1676                 },
1677                 {                               /* Target_nullfunc */
1678                         nullfunc_tmpl,          /* nf_template */
1679                         sizeof (nullfunc_tmpl), /* nf_size */
1680                 },
1681                 {                               /* Target_fillfunc */
1682                         execfill                /* ff_execfill */
1683                 },
1684                 {                               /* Target_machrel */
```

```
1685                        reloc_table,

1687                        ld_init_rel,            /* mr_init_rel */
1688                        ld_mach_eflags,         /* mr_mach_eflags */
1689                        ld_mach_make_dynamic,   /* mr_mach_make_dynamic */
1690                        ld_mach_update_odynamic, /* mr_mach_update_odynamic */
1691                        ld_calc_plt_addr,       /* mr_calc_plt_addr */
1692                        ld_perform_outreloc,    /* mr_perform_outreloc */
1693                        ld_do_activerelocs,     /* mr_do_activerelocs */
1694                        ld_add_outrel,          /* mr_add_outrel */
1695                        NULL,                   /* mr_reloc_register */
1696                        ld_reloc_local,         /* mr_reloc_local */
1697                        NULL,                   /* mr_reloc_GOTOP */
1698                        ld_reloc_TLS,           /* mr_reloc_TLS */
1699                        NULL,                   /* mr_assign_got */
1700                        ld_find_got_ndx,        /* mr_find_got_ndx */
1701                        ld_calc_got_offset,     /* mr_calc_got_offset */
1702                        ld_assign_got_ndx,      /* mr_assign_got_ndx */
1703                        ld_assign_plt_ndx,      /* mr_assign_plt_ndx */
1704                        NULL,                   /* mr_allocate_got */
1705                        ld_fillin_gotplt,       /* mr_fillin_gotplt */
1706                },
1707                {                       /* Target_machsym */
1708                        NULL,                   /* ms_reg_check */
1709                        NULL,                   /* ms_mach_sym_typecheck */
1710                        NULL,                   /* ms_is_regsym */
1711                        NULL,                   /* ms_reg_find */
1712                        NULL                    /* ms_reg_enter */
1713                }
1714        };

1716        return (&_ld_targ);
1717 }
```

```
**********************************************************
    2522 Mon Nov  5 20:40:46 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/Makefile
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

# Copyright 2012, Richard Lowe.

include $(SRC)/Makefile.master

# We have to use GCC, and only GCC.  The best way is to ask cw(1) which GCC to u
CC_CMD =        $(ONBLD_TOOLS)/bin/$(MACH)/cw -_gcc -_compiler
CC =            $(CC_CMD:sh)
CFLAGS =        -O1 -m64

LINK.c = env LD_ALTEXEC=$(PROTO)/usr/bin/amd64/ld $(CC) $(CFLAGS) -o $@ $^
COMPILE.c = $(CC) $(CFLAGS) -c -o $@ $^
COMPILE.s = $(CC) $(CFLAGS) -c -o $@ $^

.KEEP_STATE:

install default: all

.c.o:
        $(COMPILE.c)

.s.o:
        $(COMPILE.s)

# A basic use of TLS that uses the movq m/r --> movq i/r variant
PROGS +=        style2
STYLE2OBJS =    style2.o
style2: $(STYLE2OBJS)
        $(LINK.c)

# A copy of style2 that uses %r13 in the TLS sequence, and thus excercises the
# REX transitions of the movq mem,reg -> movq imm,reg variant.
PROGS +=        style2-with-r13
STYLE2R13OBJS = style2-with-r13.o
style2-with-r13: $(STYLE2R13OBJS)
        $(LINK.c)

# A copy of style2 that uses %r12 in the TLS sequence, so we can verify that
# it is _not_ special to this variant
PROGS +=        style2-with-r12
STYLE2R12OBJS = style2-with-r12.o
style2-with-r12: $(STYLE2R12OBJS)
        $(LINK.c)

# A copy of style2 that has a R_AMD64_GOTTPOFF relocation with a bad insn sequen
STYLE2BADNESSOBJS =     style2-with-badness.o
style2-with-badness: $(STYLE2BADNESSOBJS)
        -$(LINK.c)

# A basic use of TLS that uses the addq mem/reg --> leaq mem,reg variant
PROGS +=        style1
```



```
STYLE1OBJS =    style1-main.o style1-func.o
style1: $(STYLE1OBJS)
        $(LINK.c)

# A copy of style1-func that uses %r13 in the TLS sequence and thus excercises
# the REX transitions. of the addq mem,reg --> leaq mem,reg variant
PROGS +=        style1-with-r13
STYLE1R13OBJS = style1-main.o style1-func-with-r13.o
style1-with-r13: $(STYLE1R13OBJS)
        $(LINK.c)

# A copy of style1-func that uses %r12 to test the addq mem,reg --> addq imm,reg
PROGS +=        style1-with-r12
STYLE1R12OBJS = style1-main.o style1-func-with-r12.o
style1-with-r12: $(STYLE1R12OBJS)
        $(LINK.c)

all:    $(PROGS)

clobber clean:
        rm -f $(PROGS) $(STYLE1OBJS) $(STYLE1R13OBJS) $(STYLE1R12OBJS) \
            $(STYLE2OBJS) $(STYLE2R13OBJS) $(STYLE2R12OBJS) $(STYLE2BADNESSOBJS)

fail: style2-with-badness FRC

FRC:
#endif /* ! codereview */
```

****
306 Mon Nov 5 20:40:47 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/README
3337 x64 link-editor is painfully literal-minded about TLS
****

```
This tests the x64 link-editor's handling of Initial Executable TLS sequences.

The original C source files are in orig/ but unused, since we need to avoid
any changes to the compiler influencing our tests.

    % ksh test.sh /path/to/proto/root
    pass: addq-->leaq 1
    ...
    pass: bad insn sequence
#endif /* ! codereview */
```

****
567 Mon Nov 5 20:40:47 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/orig/style1-func.c
3337 x64 link-editor is painfully literal-minded about TLS
****

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

#include <stdio.h>

extern __thread char *foo, *bar;

void
func()
{
        printf("foo: %p bar: %p\n", &foo, &bar);
}
#endif /* ! codereview */
```

****
636 Mon Nov 5 20:40:47 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/orig/style1-main.c
3337 x64 link-editor is painfully literal-minded about TLS
****

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

#include <stdio.h>

extern void func();

__thread char *foo = "foo";
__thread char *bar = "bar";

int
main(void)
{
        printf("foo: %p bar: %p\n", &foo, &bar);
        func();
        return (0);
}
#endif /* ! codereview */
```

```
**********************************************************
     607 Mon Nov  5 20:40:47 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/orig/style2.c
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

#include <stdio.h>

__thread char *foo __attribute__((tls_model("initial-exec"))) = "foo";

int
main(void)
{
        printf("foo: %p\n", &foo);
        return (0);
}
#endif /* ! codereview */
```

```
**********************************************************
     847 Mon Nov  5 20:40:47 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style1-func-with-r12.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p bar: %p\n"
        .text
.globl func
        .type   func, @function
func:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movq    %fs:0, %r12
        movq    %r12, %rdx
        addq    bar@GOTTPOFF(%rip), %rdx
        addq    foo@GOTTPOFF(%rip), %r12
        movq    %r12, %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        leave
        ret
.LFE0:
        .size   func, .-func
#endif /* ! codereview */
```

```
**********************************************************
     841 Mon Nov  5 20:40:48 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style1-func-with-r13.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */
        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p bar: %p\n"
        .text
.globl func
        .type   func, @function
func:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movq    %fs:0, %r13
        movq    %r13, %rdx
        addq    bar@GOTTPOFF(%rip), %rdx
        addq    foo@GOTTPOFF(%rip), %r13
        movq    %r13, %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        leave
        ret
.LFE0:
        .size   func, .-func
#endif /* ! codereview */
```

```
**********************************************************
     815 Mon Nov  5 20:40:48 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style1-func.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p bar: %p\n"
        .text
.globl func
        .type   func, @function
func:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movq    %fs:0, %rsi
        movq    %rsi, %rdx
        addq    bar@GOTTPOFF(%rip), %rdx
        addq    foo@GOTTPOFF(%rip), %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        leave
        ret
.LFE0:
        .size   func, .-func
#endif /* ! codereview */
```

```
**********************************************************
    1114 Mon Nov  5 20:40:48 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style1-main.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p bar: %p\n"
        .text
.globl main
        .type   main, @function
main:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movq    %fs:0, %rsi
        leaq    bar@TPOFF(%rsi), %rdx
        addq    $foo@TPOFF, %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        call    func
        movl    $0, %eax
        leave
        ret
.LFE0:
        .size   main, .-main
.globl foo
        .section        .rodata.str1.1
.LC1:
        .string "foo"
        .section        .tdata,"awT",@progbits
        .align 8
        .type   foo, @object
        .size   foo, 8
foo:
        .quad   .LC1
.globl bar
        .section        .rodata.str1.1
.LC2:
        .string "bar"
        .section        .tdata
        .align 8
        .type   bar, @object
        .size   bar, 8
bar:
        .quad   .LC2
#endif /* ! codereview */
```

```
**********************************************************
     926 Mon Nov  5 20:40:48 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style2-with-badness.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p\n"
        .text
.globl main
        .type   main, @function
main:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        leaq    foo@GOTTPOFF(%rip), %rsi
        addq    %fs:0, %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        ret
.LFE0:
        .size   main, .-main
.globl foo
        .section        .rodata.str1.1
.LC1:
        .string "foo"
        .section        .tdata,"awT",@progbits
        .align 8
        .type   foo, @object
        .size   foo, 8
foo:
        .quad   .LC1
#endif /* ! codereview */
```

```
**********************************************************
     953 Mon Nov  5 20:40:48 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style2-with-r12.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
```

```
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p\n"
        .text
.globl main
        .type   main, @function
main:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movq    foo@GOTTPOFF(%rip), %r12
        addq    %fs:0, %r12
        movq    %r12, %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        ret
.LFE0:
        .size   main, .-main
.globl foo
        .section        .rodata.str1.1
.LC1:
        .string "foo"
        .section        .tdata,"awT",@progbits
        .align 8
        .type   foo, @object
        .size   foo, 8
foo:
        .quad   .LC1
#endif /* ! codereview */
```

```
**********************************************************
     953 Mon Nov  5 20:40:49 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style2-with-r13.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p\n"
        .text
.globl main
        .type   main, @function
main:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movq    foo@GOTTPOFF(%rip), %r13
        addq    %fs:0, %r13
        movq    %r13, %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        ret
.LFE0:
        .size   main, .-main
.globl foo
        .section        .rodata.str1.1
.LC1:
        .string "foo"
        .section        .tdata,"awT",@progbits
        .align 8
        .type   foo, @object
        .size   foo, 8
foo:
        .quad   .LC1
#endif /* ! codereview */
```

```
**********************************************************
     926 Mon Nov  5 20:40:49 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/style2.s
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
/*
 * This file and its contents are supplied under the terms of the
 * Common Development and Distribution License ("CDDL"), version 1.0.
 * You may only use this file in accordance with the terms of version
 * 1.0 of the CDDL.
 *
 * A full copy of the text of the CDDL should have accompanied this
 * source.  A copy of the CDDL is also available via the Internet at
 * http://www.illumos.org/license/CDDL.
 */

/*
 * Copyright 2012, Richard Lowe.
 */

        .section        .rodata.str1.1,"aMS",@progbits,1
.LC0:
        .string "foo: %p\n"
        .text
.globl main
        .type   main, @function
main:
.LFB0:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        movq    foo@GOTTPOFF(%rip), %rsi
        addq    %fs:0, %rsi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        ret
.LFE0:
        .size   main, .-main
.globl foo
        .section        .rodata.str1.1
.LC1:
        .string "foo"
        .section        .tdata,"awT",@progbits
        .align 8
        .type   foo, @object
        .size   foo, 8
foo:
        .quad   .LC1
#endif /* ! codereview */
```

```
**********************************************************
    2078 Mon Nov  5 20:40:49 2012
new/usr/src/cmd/sgs/test/ld/x64/tls/ie/x64-ie-test.sh
3337 x64 link-editor is painfully literal-minded about TLS
**********************************************************
```

```
#!/bin/ksh
#
# This file and its contents are supplied under the terms of the
# Common Development and Distribution License ("CDDL"), version 1.0.
# You may only use this file in accordance with the terms of version
# 1.0 of the CDDL.
#
# A full copy of the text of the CDDL should have accompanied this
# source.  A copy of the CDDL is also available via the Internet at
# http://www.illumos.org/license/CDDL.
#

# Copyright 2012, Richard Lowe.

function grep_test {
    name=$1
    pattern=$2

    if /usr/xpg4/bin/fgrep -q "${pattern}"; then
        print -u2 "pass: $name"
    else
        print -u2 "FAIL: $name"
    fi
}

function dis_test {
    name=${1}
    func=${2}
    file=${3}
    pattern=${4}

    dis -F${func} ${file} | grep_test "${name}" "${pattern}"
}

make PROTO="${1}"

dis_test "addq-->leaq 1" func style1 \
    'func+0x10: 48 8d 92 f8 ff ff  leaq   -0x8(%rdx),%rdx'
dis_test "addq-->leaq 2" func style1 \
    'func+0x17: 48 8d b6 f0 ff ff  leaq   -0x10(%rsi),%rsi'

dis_test "addq-->leaq w/REX 1" func style1-with-r13 \
    'func+0x10: 48 8d 92 f8 ff ff  leaq   -0x8(%rdx),%rdx'
dis_test "addq-->leaq w/REX 2" func style1-with-r13 \
    'func+0x17: 4d 8d ad f0 ff ff  leaq   -0x10(%r13),%r13'

dis_test "addq-->addq for SIB 1" func style1-with-r12 \
    'func+0x10: 48 8d 92 f8 ff ff  leaq   -0x8(%rdx),%rdx'
dis_test "addq-->addq for SIB 2" func style1-with-r12 \
    'func+0x17: 49 81 c4 f0 ff ff  addq   $-0x10,%r12   <0xfffffffffffffff0>'

dis_test "movq-->movq" main style2 \
    'main+0x4:  48 c7 c6 f0 ff ff  movq   $-0x10,%rsi   <0xfffffffffffffff0>'

dis_test "movq-->movq w/REX" main style2-with-r13 \
    'main+0x4:  49 c7 c5 f0 ff ff  movq   $-0x10,%r13   <0xfffffffffffffff0>'

dis_test "movq-->movq incase of SIB" main style2-with-r12 \
    'main+0x4:  49 c7 c4 f0 ff ff  movq   $-0x10,%r12   <0xfffffffffffffff0>'

make PROTO="${1}" fail 2>&1 | grep_test "bad insn sequence" \
```



```
    'ld: fatal: relocation error: R_AMD64_TPOFF32: file style2-with-badness.o: sy
#endif /* ! codereview */
```